

Heidi Gebauer

Juraj Hromkovič

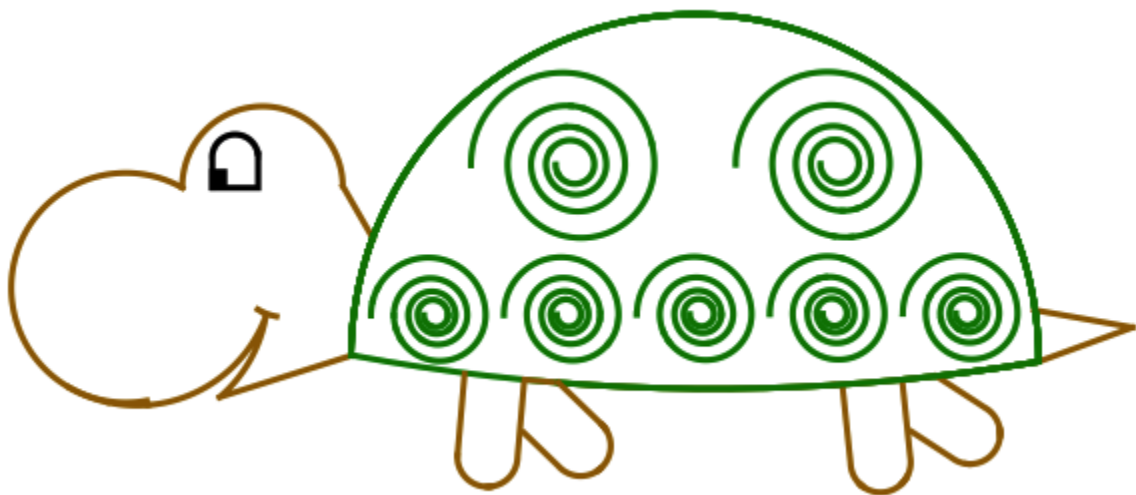
Lucia Di Caro

Ivana Kosírová

Giovanni Serafini

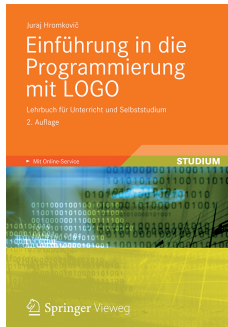
Björn Steffen

Programmieren mit LOGO



Programmare con LOGO

Questo materiale didattico 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*. 3. Aufl., Springer Vieweg 2014. ISBN: 978-3-658-04832-7.

Versione 3.3, 29 aprile 2013, SVN-Rev: 11591

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

Ambiente di sviluppo

Il materiale didattico è stato realizzato per l'ambiente di sviluppo XLogoOnline. Il programma è disponibile gratuitamente sul sito internet <http://abz.inf.ethz.ch/logo>.

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

Diritti d'uso

L'ABZ mette a disposizione questo corso gratuitamente come supporto all'insegnamento dell'informatica a docenti o istituzioni interessati per uso interno.

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 estremamente ridotto di istruzioni. Se vogliamo fargli eseguire delle attività complesse, dobbiamo istruire il computer in modo preciso, 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:



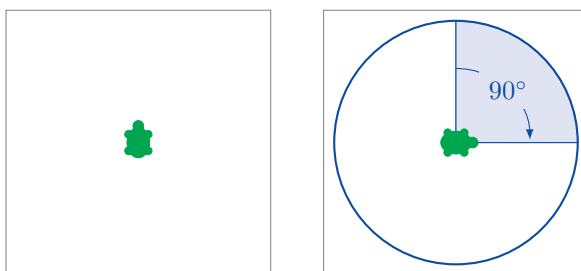
Cancellare tutto e ricominciare da capo

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

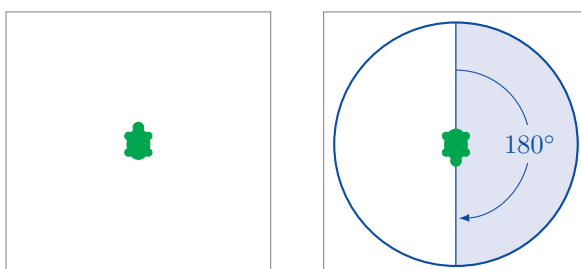
Girare a destra e a sinistra

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

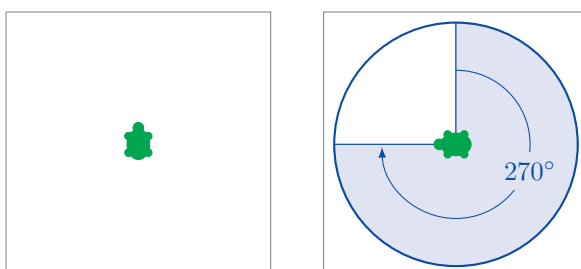
Con l'istruzione **right 90** oppure **rt 90** la tartaruga si gira di 90° verso destra. Ciò corrisponde a un quarto di cerchio:



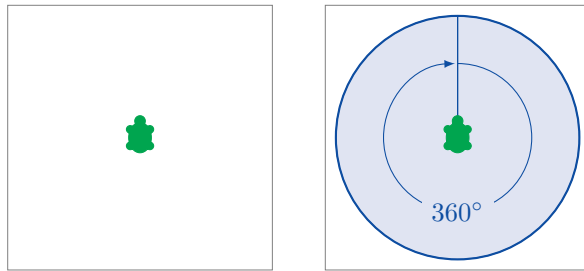
L'istruzione **right 180** oppure **rt 180** fa girare la tartaruga di 180° verso destra. Ciò corrisponde a un semicerchio:



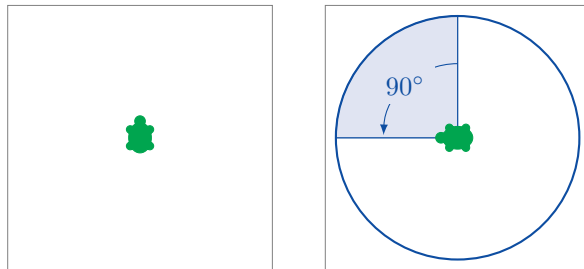
right 270 oppure **rt 270** fa girare la tartaruga di 270° verso destra:



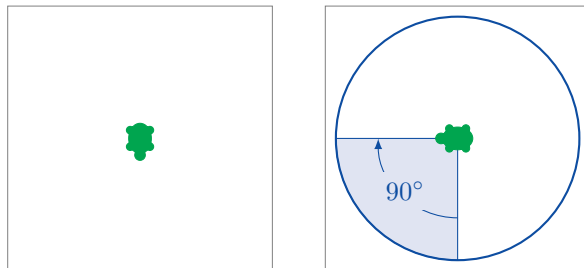
Le istruzioni **right 360** e **rt 360** fanno girare la tartaruga di 360° verso destra. La tartaruga fa un giro completo su se stessa.



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

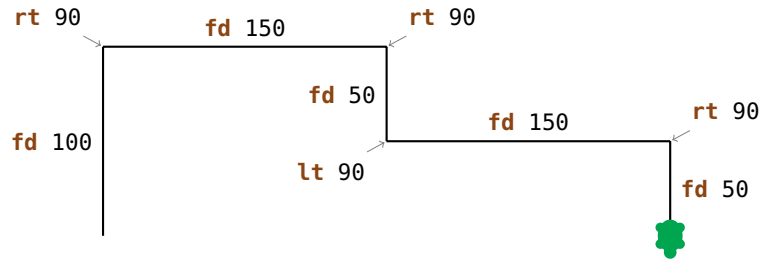
Programmare significa scrivere una sequenza di istruzioni.

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:

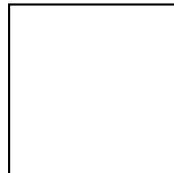
```
fd 100  
rt 90  
fd 200  
rt 90  
fd 80  
rt 90  
fd 100  
rt 90  
fd 50
```



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

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 spiegare 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]
comando per ripetere una sequenza di istruzioni	numero di ripetizioni	sequenza di istru- zioni 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 ad abbreviarlo facendo capo all'istruzione **repeat**?

Esercizio 7

Immetti 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 ad abbreviarlo facendo capo all'istruzione **repeat**?

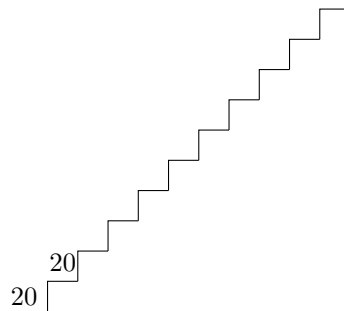
Possiamo risolvere molti esercizi utilizzando proprio quest'ultima strategia. Per prima cosa, devi sempre scoprire il motivo che si ripete. In seguito devi scrivere due programmi: un programma per il *motivo* e un altro per il *riposizionamento* della tartaruga in modo che sia pronta a disegnare nuovamente motivo. Il tuo programma avrà la struttura seguente:

repeat *numero* [*motivo riposizionamento*]

Esercizio 10

Disegniamo 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.

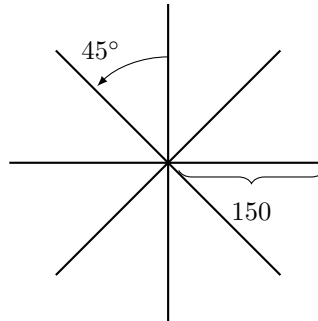
(b) Disegna una scala con 5 gradini di grandezza 50.

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

Esercizio 11

Adesso vogliamo disegnare delle stelle.

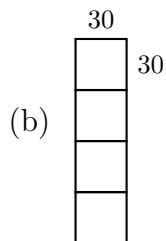
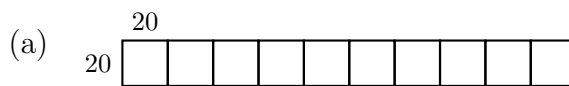
(a) Disegna la stella seguente:



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

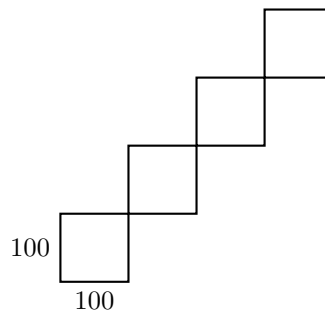
Esercizio 12

Disegna le figure seguenti:



Esercizio 13

Scrivi un programma per disegnare l'immagine seguente:



Esercizio 14

Copia ed esegui il programma che segue:

```
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

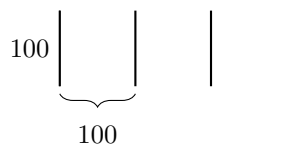
penup oppure semplicemente **pu**.

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

pendown oppure semplicemente **pd**.

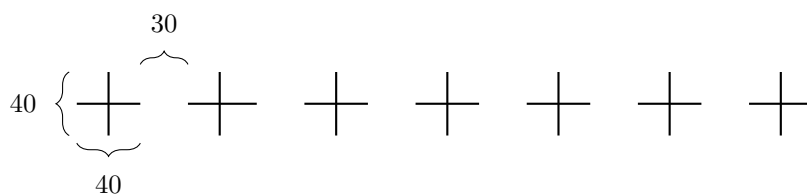
Esercizio 15

Disegna la figura seguente con l'aiuto di un programma:



Esercizio 16

Scrivi un programma per disegnare la figura seguente:



3 Se diamo un nome ad un programma, possiamo riutilizzarlo

Possiamo dare un nome ad ogni programma che scriviamo. Quando, più in là durante la lezione, scriveremo questo nome nella riga di comando, il programma verrà eseguito.

Il programma per disegnare un quadrato di lato 100 è:

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

A questo programma possiamo dare il nome **QUADRATO100**:

```
to QUADRATO100
repeat 4 [fd 100 rt 90]
end
```

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

I programmi vanno sempre essere scritti nel **blocco note**. Nel nostro nostro quaderno, tali programmi sono contrassegnati con un rettangolo su sfondo grigio.

Ognuno di noi può scegliere liberamente che nome assegnare ad un programma. In questo caso, abbiamo scelto **QUADRATO100** per sottolineare che il programma disegna un quadrato di lato 100. Il nome di un programma è composto soltanto di lettere e numeri, ma non contiene 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

QUADRATO100

nella riga di comando, verranno eseguite le istruzioni **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 QUADRATO20
repeat 4 [fd 20 rt 90]
end
```

Dopo aver disegnato **QUADRATO20** la tartaruga è posizionata nell'angolo del quadrato in basso a sinistra:



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

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

Anche in questo caso diamo un nome al nostro nuovo 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 [QUADRATO20 RIPOSIZIONARE20]
```

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

```
to FILA10
repeat 10 [QUADRATO20 RIPOSIZIONARE20]
end
```

In questo caso diciamo che i programmi **QUADRATO20** 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

```
repeat 4 [QUADRATO30 RIPOSIZIONARE30]
```

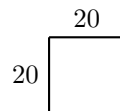
Devi quindi scrivere i sottoprogrammi **QUADRATO30** e **RIPOSIZIONARE30** adeguati.

Esercizio 18

Utilizza il programma **QUADRATO100** 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:

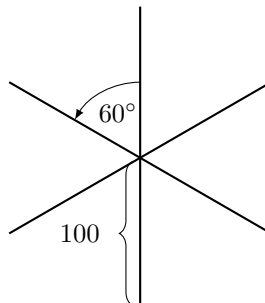
```
to LINEA  
fd 150 bk 150  
end
```

Esercizio 21

Copia ed esegui il 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

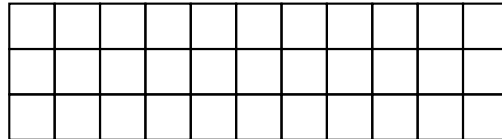
Precedentemente abbiamo scritto il programma **FILA10**. Che cosa fa il programma seguente?

```
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 dimensioni diverse tra loro.

- (a) Scrivi un programma che disegna un quadrato di lato 50 e dagli il nome **QUADRAT050**. Eseguilo per verificarne il funzionamento.
- (b) Scrivi un programma che disegna un quadrato di lato 75.
- (c) Esegui il programma

```
QUADRAT050  
QUADRAT075  
QUADRAT0100
```

Che cosa compare sullo schermo?

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

Costruiamo tante 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
```

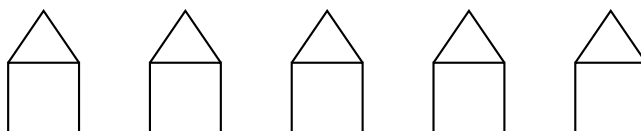
Il programma disegna la casa sottostante:



Esercizio 26

Da dove parte la tartaruga per disegnare la casa? Quale strade percorre la tartaruga mentre esegue il programma **CASA**? Dove si trova la tartaruga quando ha terminato di disegnare la casa? Disegna l'immagine corrispondente e indica, 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. Dice alla tartaruga di disegnare la prima casa, sulla sinistra, e poi di andare al punto di partenza della prossima casa:



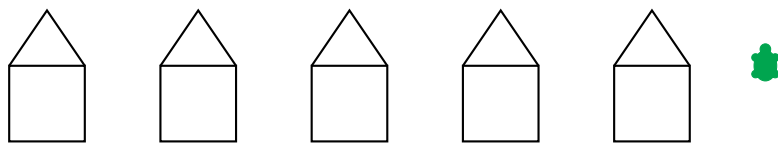
Per fare ciò, l'architetto usa il programma seguente:

```
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**:

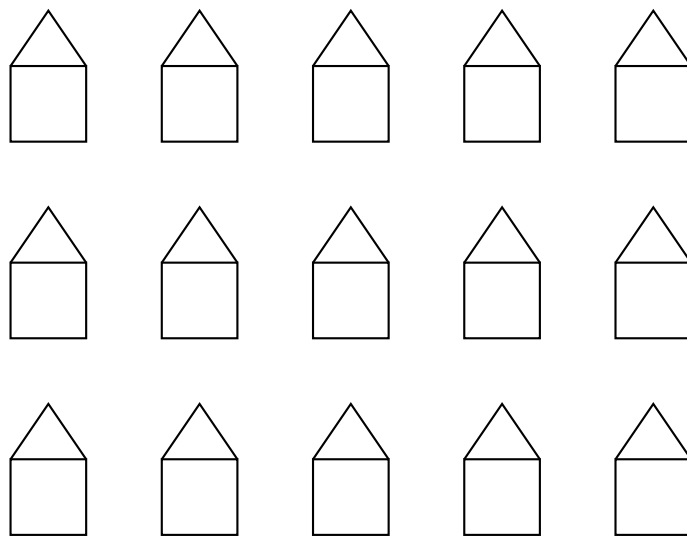
```
to FILACASE  
repeat 5 [CASA rt 90 pu fd 50 lt 90 pd]  
end
```

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



Esercizio 27

Aggiungiamo un paio file di case al nostro quartiere. Utilizza il programma **FILACASE** come motivo per disegnare l'immagine seguente:



Suggerimento: Dopo ogni fila, la tartaruga deve spostarsi e raggiungere la posizione giusta per disegnare la prossima fila.

Linee spesse e quadrati neri

Esercizio 28

Disegniamo delle linee spesse col programma **LINEASPESSA**.

Copia il programma sottostante nel blocco note, e dagli il nome **LINEASPESSA**.

```
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 100 volte il programma **LINEASPESSA** con il programma

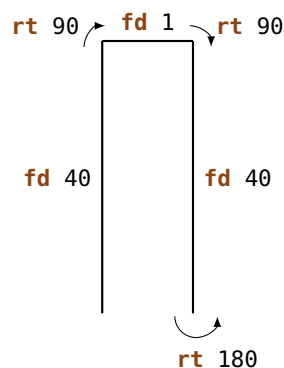
```
repeat 100 [LINEASPESSA]
```

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
```



Per costruire una linea spessa, disegniamo due linee talmente vicine l'una all'altra, da farci credere che sia una sola linea, spessa.

Scrivi ed esegui il programma **LINEASPESSA40**.

Esercizio 31

Una linea spessa di lunghezza 40 può essere interpretata come un rettangolo di base 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 40 volte `LINEASPESSA40`, otteniamo il quadrato nero di lato 40. Esegui `LINEASPESSA40` per 40 volte per verificare che l'idea funziona.

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

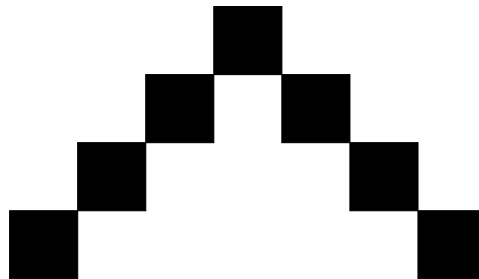
Esercizio 32

Disegna l'immagine seguente facendo capo al programma `NERO40`:



Esercizio 33

Utilizza il programma `NERO40` per disegnare l'immagine che segue:



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 **MURA**. Scrivi per l'architetto due programmi che disegnano queste due componenti ed uniscili nel programma **CASA1** per disegnare una nuova casa.

Esercizio 37

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

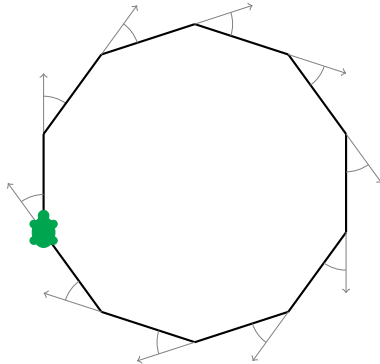
4 Poligoni regolari e cerchi

Poligoni regolari

Un poligono regolare ha n angoli identici e n lati, tutti della stessa lunghezza. Ad esempio, per disegnare un decagono (un poligono regolare di 10 lati) a matita, devi disegnare 10 linee, cambiando *un po'* la direzione dopo ogni linea. La tartaruga ruota ogni volta di un piccolo angolo.

Quanto è grande questo angolo?

Per disegnare un poligono regolare dobbiamo girare la tartaruga più volte, ma alla fine essa si ritroverà di nuovo alla posizione di partenza. Il suo sguardo sarà rivolto verso l'alto, come all'inizio.



Ciò significa che la tartaruga si è girata in tutto di 360° . Per disegnare un decagono sono necessarie 10 rotazioni, tutte dello stesso angolo. Questo angolo misura

$$\frac{360^\circ}{10} = 36^\circ$$

La tartaruga dovrà quindi girarsi di 36° verso destra: **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 l'angolo esatto, scrivendo

```
360/7
```

(/ per il computer significa *dividere*). Il computer troverà il risultato esatto. Perciò, per disegnare un ettagono di lato 100, procediamo nel modo seguente:

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

Verifica che il programma funzioni.

Disegniamo dei cerchi

Con i comandi **fd** e **rt** non è possibile disegnare dei cerchi perfetti. Ma come avrai sicuramente già notato, un poligono regolare con molti angoli assomiglia parecchio a un cerchio. Possiamo quindi ottenere un cerchio disegnando un poligono regolare con 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 si disegna un cerchio piccolissimo? Scrivi un programma adatto.
- (b) Come si disegna un cerchio grande? Scrivi un programma adatto.

Esercizio 41

Prova a disegnare i seguenti semicerchi. Puoi sceglierne liberamente le dimensioni.



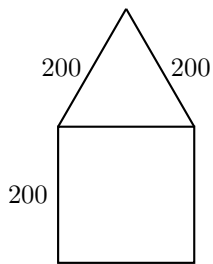
(a)



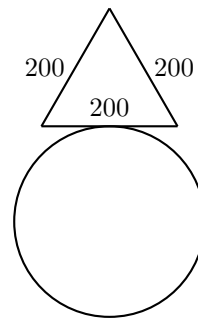
(b)

Esercizio 42

Utilizza le tue nuove conoscenze per disegnare le immagini seguenti. Scegli liberamente le dimensioni del cerchio.



(a)



(b)

Diamo spazio alla fantasia

Disegna un ettagono (poligono regolare con 7 lati):

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

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

```
rt 10
```

Ripeti un paio di volte questi due programmi e osserva l'immagine che ottieni. Dopo aver disegnato ogni ettagono, la tartaruga si gira di 10° con **rt 10**. Se vogliamo che essa torni alla posizione iniziale dobbiamo ripetere il procedimento

$$\frac{360^\circ}{10^\circ} = 36$$

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 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 che lo risolva.

Colori

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

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

Con l'istruzione

setpencolor	X
istruzione per cambiare colore	numero del colore desiderato

la tartaruga passerà ad usare il colore numero **X**. Abbreviamo l'istruzione **setpencolor** con **setpc**.

Grazie ai colori, siamo in grado di disegnare forme divertenti, come per esempio quella creata dal programma seguente. Dapprima diamo un nome a due programmi che disegnano due cerchi di dimensioni diverse tra loro:

```
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
```

Ed infine 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
```

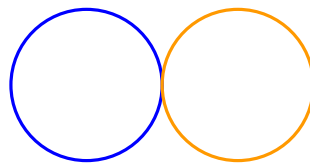
Se lo desideri, puoi continuare il lavoro intrapreso e puoi disegnare ancor di più. Oppure dà spazio alla tua fantasia ed inventa un forma che ti piaccia.

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**?

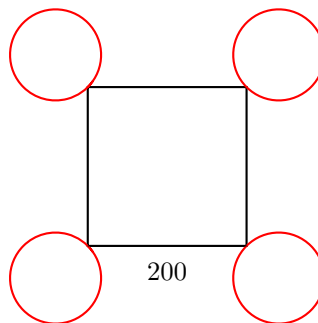
Esercizio 46

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



Esercizio 47

Scrivi un programma per disegnare l'immagine seguente. Lei libero di scegliere le dimensioni dei cerchi.



5 Programmi con parametri

Nella Lezione 3 abbiamo imparato a dare un nome ai programmi e a chiamarli per nome per far sì che la tartaruga disegni l'immagine desiderata. Nella Lezione 4 abbiamo appreso come disegnare dei poligoni regolari. Tuttavia, dover scrivere un nuovo programma per ogni poligono regolare a dipendenza del numero degli angoli è un'attività molto ripetitiva.

Studiamo, ad 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. Le uniche differenze sono i numeri (evidenziati in giallo) **7**, **12** e **18** che determinano il numero di angoli. A questo punto, vorremmo scrivere un programma con cui sia possibile disegnare un poligono regolare qualunque:

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

Che cosa abbiamo fatto? Nel nostro 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 preciso di angoli solo in un secondo tempo, dobbiamo scrivere **ANGOLI** (preceduto da **:**) anche dopo il nome del programma. Se ora scriviamo l'istruzione **POLIGONO 12** nella riga di comando, il computer sostituirà dappertutto nel programma **:ANGOLI** con il numero **12**,

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

disegnando così un dodecagono.

Prova ad eseguire

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

L'informazione **:ANGOLI** è un **parametro**. Nell'esempio soprastante, 3, 4, 5 e 6 sono i **valori del parametro :ANGOLI**. Il computer riconosce un parametro grazie al segno **:** e questa è la ragione per cui è necessario scrivere sempre **:** 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 interpretare 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 con lati di lunghezza qualsiasi:

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

Esercizio 49

I programmi seguenti disegnano cerchi di dimensioni diverse 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 dimensioni 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. Non dimenticare di scrivere sempre i due punti prima del nome 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 lato qualsiasi. In seguito utilizza questo programma per disegnare, uno dopo l'altro, dei triangoli di lato

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

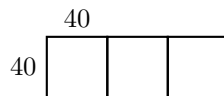
Che cosa compare sullo schermo?

Esercizio 52

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

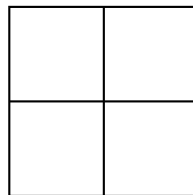


Ed ecco cosa appare se eseguiamo **QUADRATI 3**:



Esercizio 53

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

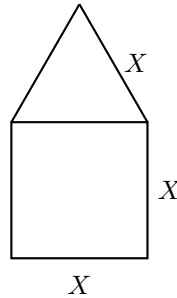


Esercizio 54

Scrivi un programma con un parametro che disegni degli esagoni 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 dimensioni qualsiasi, come nell'immagine seguente.



Programmi con più di un parametro

Un programma può avere anche più di un solo parametro. Se vogliamo disegnare dei poligoni regolari di lato qualsiasi, possiamo ad esempio definire un parametro `:ANGOLI` per il numero di angoli ed un parametro `:LATO` per la lunghezza dei lati.

Nei programmi seguenti il parametro `:ANGOLI` è evidenziato in giallo mentre il parametro `:LATO` viene evidenziato 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 regolari che vogliamo:

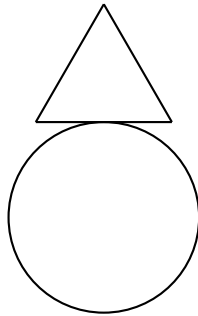
```
to POLIGONI :ANGOLI :LATO
repeat :ANGOLI [fd :LATO rt 360/:ANGOLI]
end
```

Verifica il funzionamento del programma `POLIGONI` nel modo seguente:

```
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 che disegni l'immagine seguente. Le dimensioni del cerchio e quelle 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
```

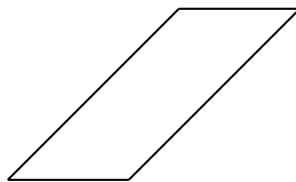
disegna un rettangolo di altezza 100 e base 200. Verificane dapprima il corretto funzionamento, e scrivi di seguito un programma con due parametri, che disegni un rettangolo di base ed altezza qualsiasi.

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 parallelogramma identico a questo, ma 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
```

ed 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 ...
```

Smetti di disegnare non appena la tartaruga si trova di nuovo alla posizione dalla quale è partita. La tartaruga avrà disegnato 18 cerchi, girandosi di 20° dopo ciascuno di essi, per una rotazione totale di $18 \times 20^\circ = 360^\circ$.

Riassumiamo: abbiamo ottenuto il programma

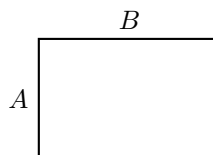
```
repeat 18 [POLIGONI 360 2 rt 20]
```

Esegui il programma, prima di rispondere alle domande seguenti.

- In modo molto simile, possiamo disegnare dei fiori con 10 oppure 20 petali (cerchi). Come si fa? Scrivi un programma adatto e verifica che funzioni correttamente.
- Riesci a scrivere un programma con un parametro per disegnare dei fiori con un numero qualsiasi di petali (cerchi)?
- Riesci a scrivere un programma con i parametri seguenti?
 - il numero di petali (cerchi) e
 - dimensioni dei cerchi

Esercizio 60

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



Oltre ai lati A e B , devi poter scegliere anche il colore.

6 Disegniamo dei fiori e passiamo dei parametri ad un sottoprogramma

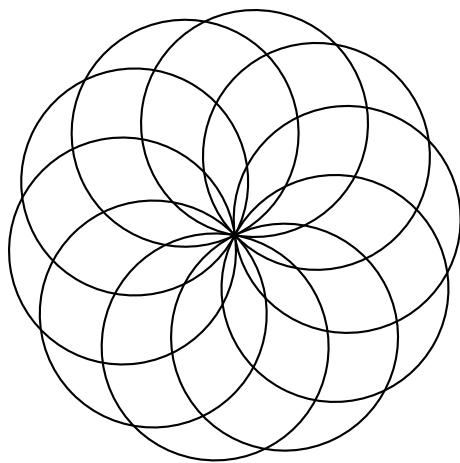
In questa lezione impariamo a disegnare dei fiori. Con l'aiuto dei parametri, cambieremo a nostro piacimento sia la forma, che il colore dei fiori. La tartaruga disegnerà figure belle, variopinte ed estremamente fantasiose.

Diamo dapprima un'occhiata al programma seguente, che copiamo nell'editore:

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

Per disegnare un fiore con 10 petali possiamo ora scrivere 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. Le dimensioni dei petali devono essere doppie rispetto a quelle dei petali disegnati nell' esercizio precedente.

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

Scrivi nell'editore:

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

Esegui **FIORE 1**, **FIORE 2**, **FIORE 3** e osserva attentamente la figura che viene disegnata. Cosa è successo? Quando abbiamo eseguito **FIORE 1**, abbiamo assegnato il valore 1 al parametro **:DI**, e al momento in cui la tartaruga esegue il sottoprogramma **CERCHI :DI**, passiamo al sottoprogramma il valore 1. Di fatto, eseguiamo **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.

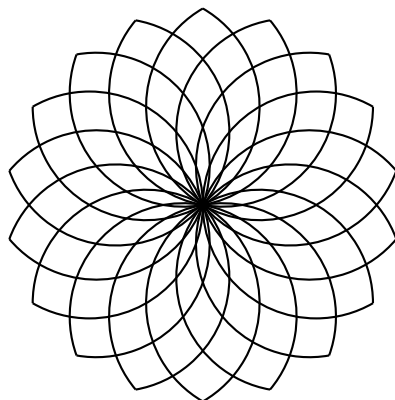
```
to FIORI :DI1 :DI2
setpc 3 FIORE :DI1
setpc 4 FIORE :DI2
end
```

Esercizio 65

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

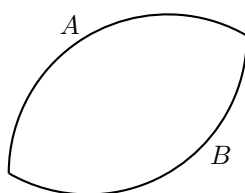
Un fiore con petali aguzzi

Vuoi imparare a disegnare un fiore con dei petali molto 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



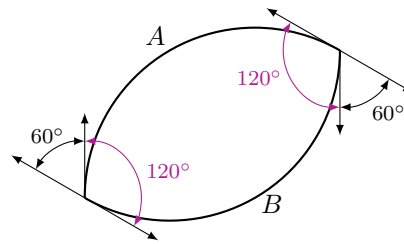
incolliamo due archi di cerchio *A* e *B*. Possiamo disegnare un arco di cerchio, ad esempio, con il programma che segue:

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

Provalo.

Ci accorgiamo immediatamente 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 dobbiamo girare la tartaruga prima di cominciare a disegnare l'arco di cerchio B per la parte inferiore del petalo? Studiamo attentamente l'immagine che segue:



Una volta terminato di disegnare, vogliamo tornare alla posizione di partenza. In tutto dobbiamo girare la tartaruga di 360° . Sia nella parte A , che nella parte B giriamo la tartaruga di 120° . Quindi rimangono ancora

$$360^\circ - 120^\circ - 120^\circ = 120^\circ$$

da distribuire in parti uguali tra le due rotazioni, una volta raggiunte le punte del petalo:

$$\frac{120^\circ}{2} = 60^\circ.$$

Il risultato è il programma che segue:

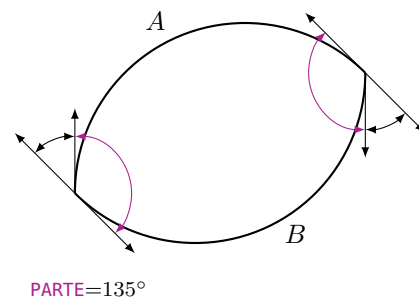
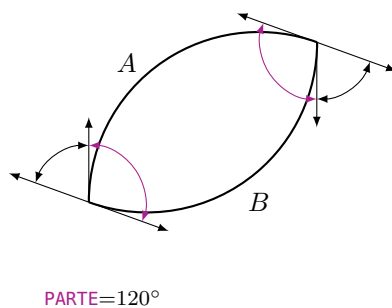
```
repeat 120 [fd 2 rt 1]
rt 60
repeat 120 [fd 2 rt 1]
rt 60
```

oppure più semplicemente:

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

Provalo.

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



Anche in questa situazione possiamo fare capo ad un parametro, a cui diamo, ad esempio, il nome **: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 $\frac{360^\circ}{2} = 180^\circ$. Perciò la rotazione sulla punta del petalo è

$$180^\circ - \text{:PARTE}.$$

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

```
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?

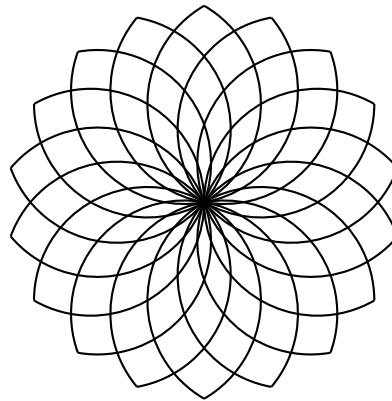
Un fiore è composto da tanti petali aguzzi

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

Esercizio 66

Disegna dapprima un fiore con il 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?

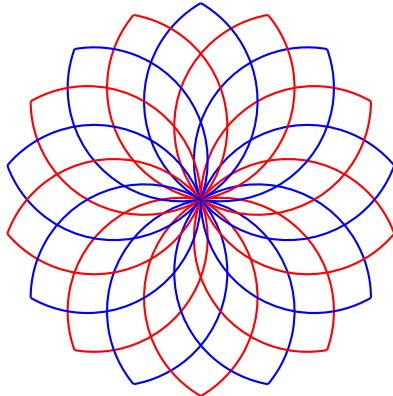
Scrivi il programma per disegnare il fiore in una riga sola utilizzando in modo adatto l'istruzione **repeat**. (Ricorda che la somma delle rotazioni **rt** tra i singoli petali deve dare in tutto 360° .)

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 **:DI** (dimensioni). Scrivi il programma

```
PETALI :PARTE :DI
```

con i parametri **:PARTE** e **:DI**, che ci permettono di scegliere l'arco di cerchio e le dimensioni 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
```

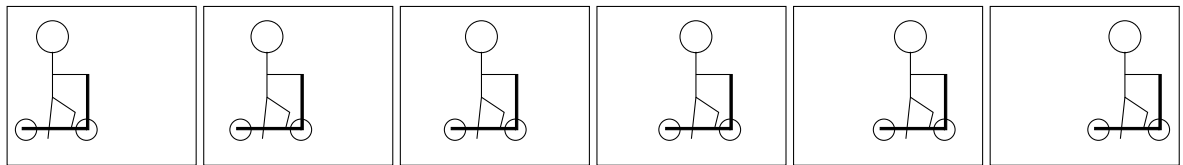
Gira poi la tartaruga verso destra di 80° e ripeti il programma soprastante.

Esercizio 70

Inventa altre figure fantasiose.

7 Programmiamo delle animazioni

Sai come si prepara un cartone animato? Si procede come in un libretto animato. Dapprima 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 poco tra un'immagine e la successiva:



Se mettiamo le immagini una sopra l'altra e le sfogliamo velocemente col pollice, abbiamo l'impressione che il ragazzo si sposti sul suo monopattino da sinistra verso destra. Le **animazioni** non sono altro che immagini in movimento.

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

Un quadrato che lascia delle tracce

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



Il programma che segue, come ben sappiamo, ci permette di disegnare un quadrato:

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

Dopo aver disegnato il quadrato una prima volta, spostiamo la tartaruga un po' verso destra e disegniamo il quadrato una seconda volta. Di seguito, spostiamo la tartaruga di nuovo verso destra e disegniamo nuovamente il quadrato. Ripetiamo questo procedimento più volte.

Grazie al programma seguente, disegniamo 120 volte il nostro quadrato:

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

Esercizio 71

Scrivi i programmi **QUAD100** e **QUADANIM** nell'editore ed esegui **QUADANIM**. Che cosa appare sullo schermo?

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

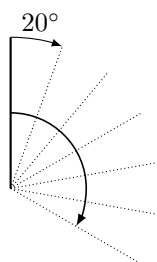


Esercizio 72

Fai muovere il quadrato dal basso verso l'altro invece che da sinistra verso 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:



Come disegnare e cancellare un quadrato

Per far sparire tracce fastidiose, dobbiamo imparare a cancellare le figure dopo averle disegnate. A tal fine, la tartaruga deve utilizzare una gomma invece di una matita. Con l'istruzione **penerase** (oppure, in modo più compatto, **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? Dobbiamo dirglielo, con l'istruzione `penpaint` (oppure più semplicemente `ppt`). Utilizziamo subito questa nuova istruzione nel 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 spiegarcelo?

Il quadrato deve pazientare un po'

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

Procediamo come segue:

<code>wait</code>	4
istruzione	tempo di attesa
per aspettare	

Esercizio 76

Prova il programma

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

Un quadrato che si sposta da sinistra verso destra

Adesso siamo finalmente pronti ad 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 questo ultimo programma. Se durante l'animazione la tartaruga ti dà fastidio, la puoi nascondere. Inserisci l'istruzione **hideturtle** (oppure, in modo più compatto, **ht**) all'inizio del programma. Ti accorgerai che l'animazione diventerà più veloce. Inserisci l'istruzione **showturtle** (oppure, in modo più compatto, **st**) alla fine del programma, prima di **end**. Così facendo la tartaruga apparirà di nuovo.

Esercizio 77

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

Esercizio 78

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

Esercizio 79

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

Esercizio 80

Modifica il programma **QUADANIM** in modo tale che il quadrato si sposti da destra verso sinistra invece che da sinistra verso destra.

Esercizio 81

Rifletti sul programma seguente ed in seguito esegilo per verificare la bontà della 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 ed in seguito esegilo per verificare la bontà della 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

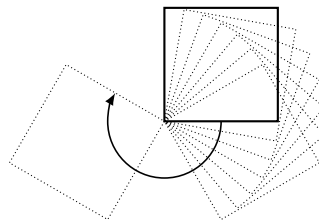
Fai capo al programma seguente

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

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

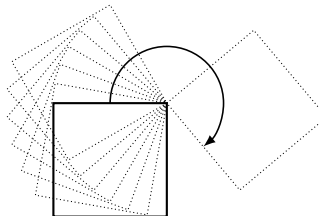
Esercizio 86

Fai ruotare un quadrato in senso orario attorno al vertice inferiore sinistro. Puoi scegliere liberamente la lunghezza del lato:



Esercizio 87

Ora fai ruotare il quadrato in senso orario attorno al suo vertice superiore destro:



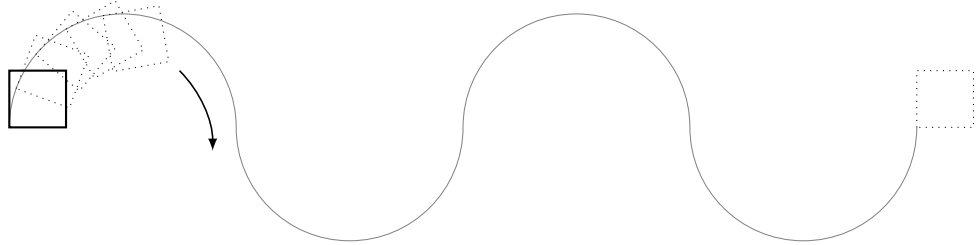
Se già sai come utilizzare i parametri, puoi risolvere gli esercizi seguenti.

Esercizio 88

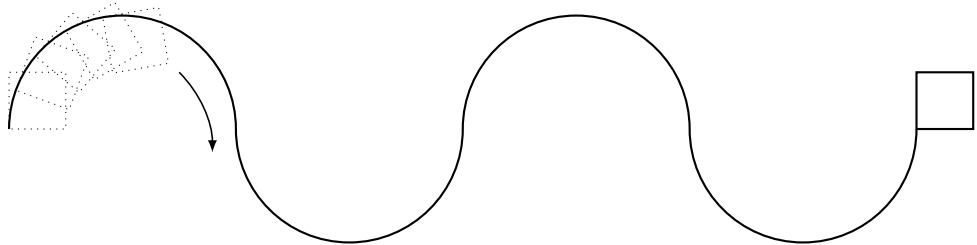
Scrivi un programma con *due parametri* per far muovere un quadrato da sinistra verso destra. Il primo parametro deve determinare la lunghezza del lato, il secondo parametro la velocità con cui si muove il quadrato.

Esercizio 89

- (a) Fai muovere un quadrato lungo la traiettoria composta da 4 semicerchi che viene rappresentata di seguito. La lunghezza del lato del quadrato deve essere determinata da un parametro.

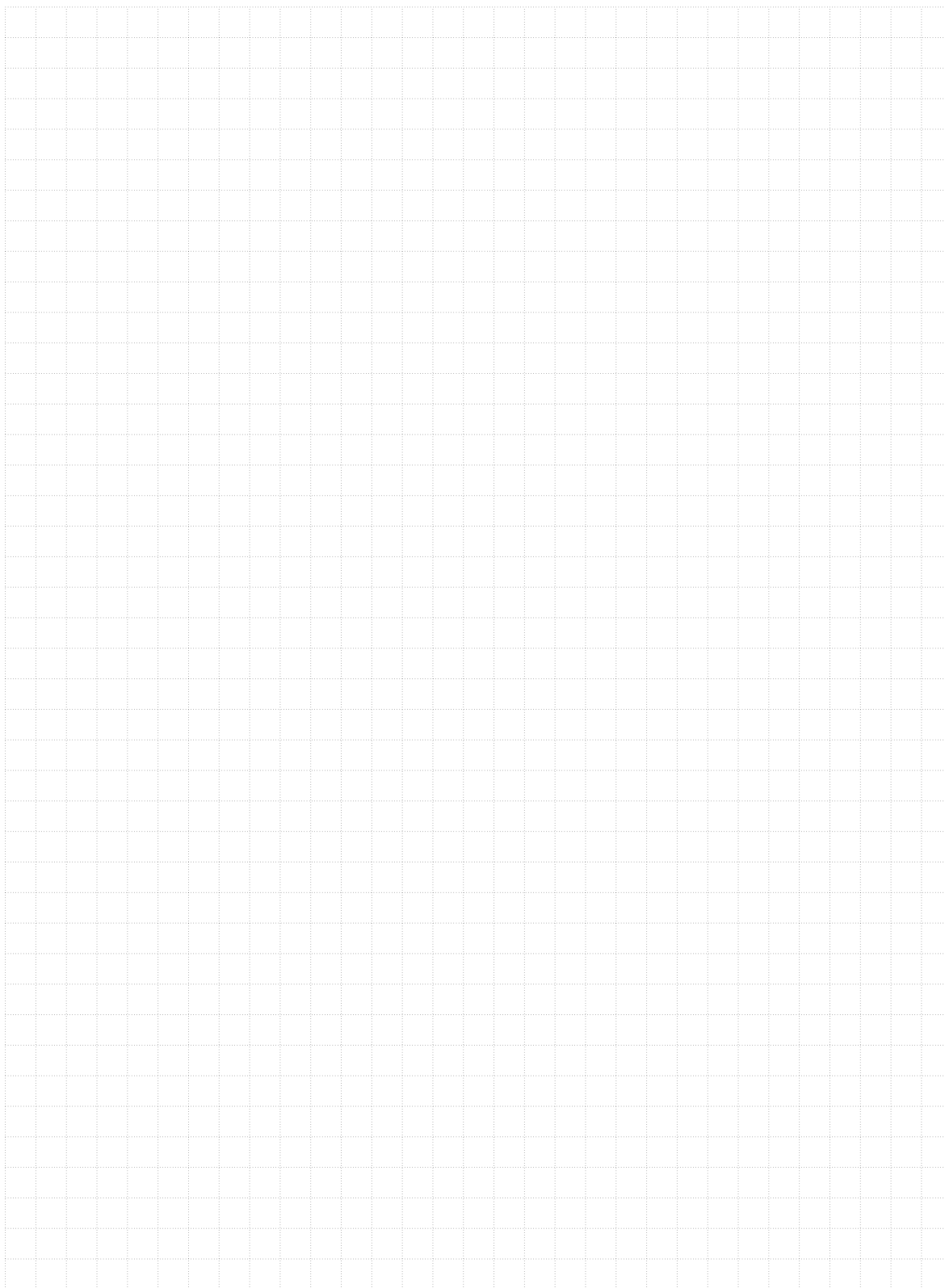


- (b) Ora vogliamo che anche la traiettoria venga disegnata con una traccia che appare man mano che il quadrato si sposta.



- (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



Lista delle istruzioni

- fd** 100 vai 100 passi in avanti
- bk** 50 vai 50 passi indietro
- cs** cancella tutto e ricomincia da capo
- rt** 90 gira di 90 gradi a destra
- lt** 90 gira 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 prendi la matita con il colore numero 3
- to** NOME crea un programma con un nome
- to** NOME :PARAMETRO crea un programma con un nome ed un parametro
 - end** tutti i programmi con un nome terminano con questa istruzione
 - pe** la tartaruga prende in mano la gomma
 - ppt** la tartaruga prende in mano la matita
 - wait** 5 la tartaruga aspetta 5 unità di tempo



Programmieren mit LOGO

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

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