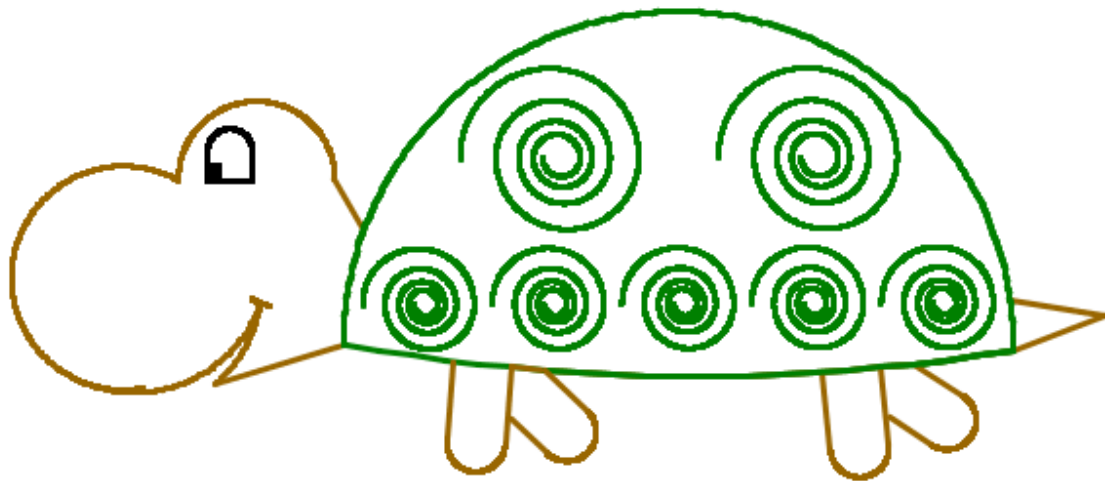


Heidi Gebauer Juraj Hromkovič Lucia Keller
Ivana Kosírová Giovanni Serafini Björn Steffen

Programmer avec LOGO

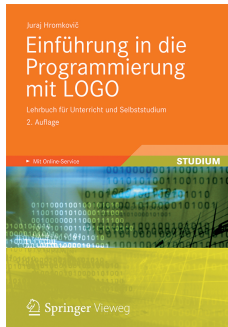


ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Programmer avec LOGO

Ce script est une version allégée des leçons 1–7 du manuel *Einführung in die Programmierung mit LOGO* (en allemand). Le manuel contient de nombreux exercices et explications supplémentaires. Il comporte également des indications destinées aux enseignants. Le manuel comprend 15 leçons.



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

Version 3.0, 1^{er} mai 2013, SVN-Rev: 11616

Traduction en Français: Alain Vaucher
Lectorat: Ivana Kosírová, Giovanni Serafini

Environnement de programmation

Ce manuel d'enseignement a été réalisé pour l'environnement de programmation XLogo. XLogo est disponible gratuitement sur le site internet xlogo.tuxfamily.org.

XLogo doit être configuré en anglais afin que les programmes Logo présents dans ce manuel puissent être exécutés tels quels.

Droits d'utilisation

Ce manuel a été conçu pour encourager l'enseignement de l'informatique. L'ABZ le met gratuitement à disposition des enseignants et institutions intéressés, dans le cadre d'une utilisation interne.

ABZ

L'ABZ (Ausbildungs- und Beratungszentrum für Informatikunterricht der ETH Zürich) est le Centre de formation et de consultation pour l'enseignement de l'informatique de l'EPF de Zurich. Il soutient, au moyen d'une offre variée, les écoles et enseignants qui souhaitent mettre en place et développer l'enseignement de l'informatique. Cette offre comprend entre autres des conseils individuels, l'enseignement par des professeurs de l'ETH et par l'équipe de l'ABZ sur place dans les écoles, des cours de formation continue pour les enseignants ainsi que du matériel d'enseignement.

www.abz.inf.ethz.ch

1 Instructions de base

Une **instruction d'ordinateur** est une directive qui peut être comprise et exécutée par l'ordinateur. En fait, l'ordinateur ne comprend que peu d'instructions, c'est pourquoi nous devons combiner des instructions d'ordinateur pour effectuer des tâches complexes. Cette suite d'instructions simples est appelée **programme**. Il n'est pas toujours facile d'écrire des programmes. Il existe certains programmes, qui sont composés de plusieurs millions d'instructions. Afin de ne pas perdre la vue d'ensemble, il est nécessaire de procéder d'une manière propre et méthodique. C'est ceci que nous voulons apprendre dans ce cours de programmation.

Dessiner des lignes droites

Avec l'instruction **forward 100** ou **fd 100**, tu fais avancer la tortue de 100 pas vers l'avant:



Avec l'instruction **back 100** ou **bk 100**, la tortue recule de 100 pas:



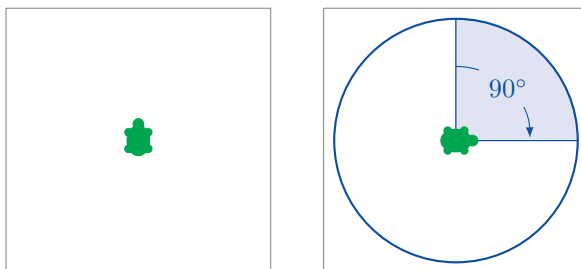
Effacer et recommencer

L'instruction **cs** efface tous les traits dessinés et la tortue retourne à sa position initiale.

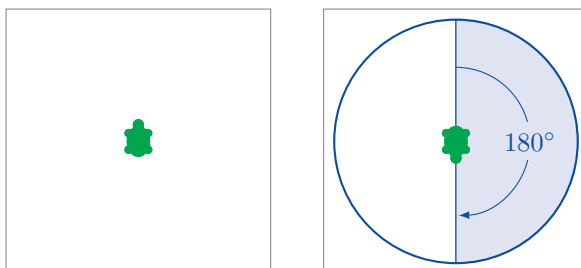
Tourner

La tortue ne peut avancer que droit devant elle, c'est pourquoi il existe des instructions pour la faire tourner.

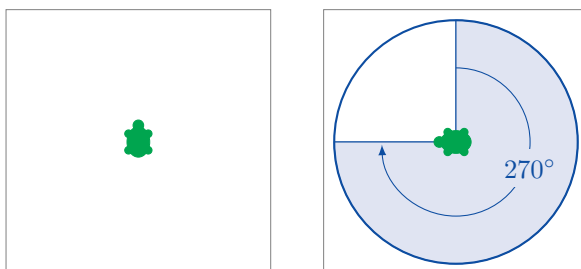
Avec l'instruction **right 90** ou **rt 90**, la tortue tourne sur elle-même de 90° vers la droite, ce qui correspond à un quart de tour:



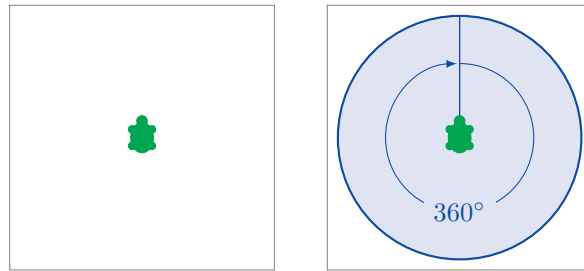
L'instruction **right 180** ou **rt 180** fait pivoter la tortue de 180° vers la droite. Ceci correspond à un demi-tour:



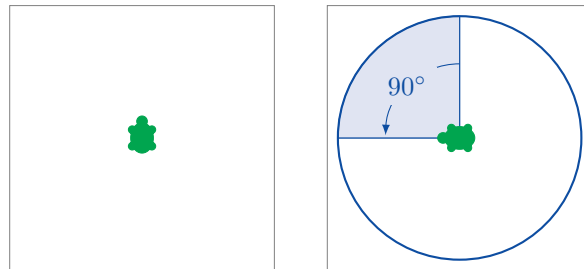
right 270 ou **rt 270** tourne la tortue de 270° vers la droite:



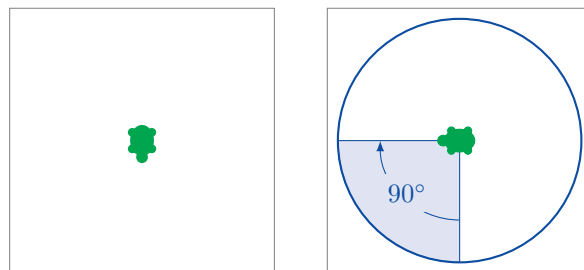
Les instructions **right 360** et **rt 360** tournent la tortue de 360° vers la droite, ce qui correspond à un tour entier:



Avec l'instruction **left 90** ou **lt 90**, la tortue tourne de 90° vers la gauche:



La rotation est toujours effectuée par rapport au point de vue de la tortue, comme le montre l'exemple suivant avec l'instruction **rt 90**:



Programmer

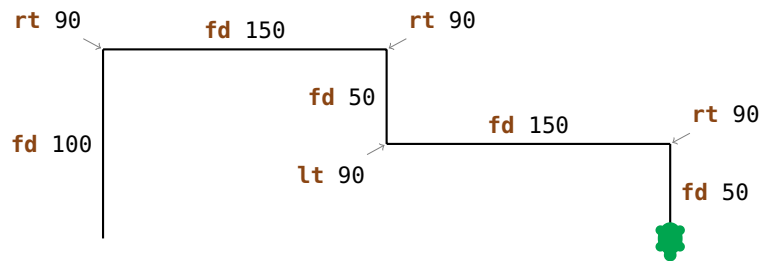
Programmer signifie que l'on écrit plusieurs instructions d'ordinateur l'une après l'autre.

Exercice 1

Copie le programme suivant et exécute-le:

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

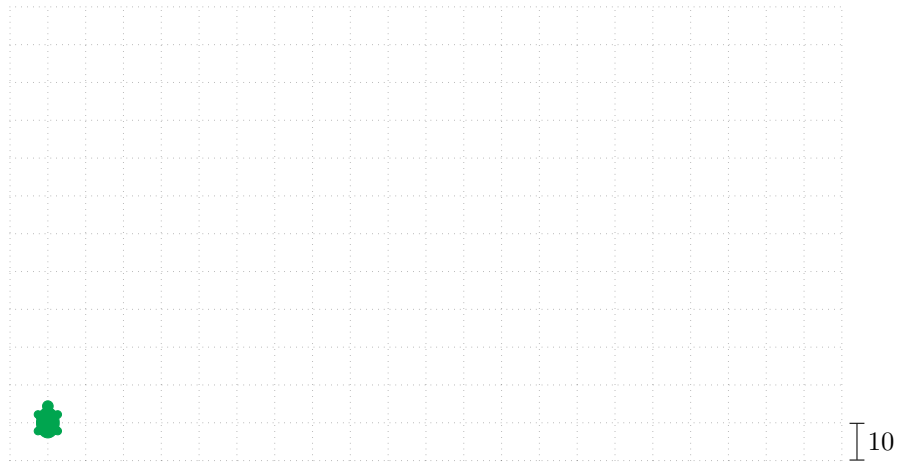
As-tu dessiné l'image suivante?



Exercice 2

Écris le programme suivant et exécute-le:

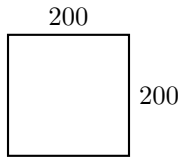
```
fd 100  
rt 90  
fd 200  
rt 90  
fd 80  
rt 90  
fd 100  
rt 90  
fd 50
```



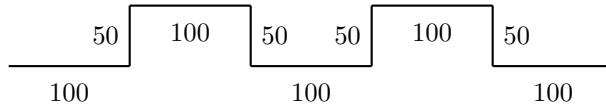
Dessine l'image créée à côté du programme ci-dessus et décris (comme dans l'Exercice 1) ce que fait chaque instruction.

Exercice 3

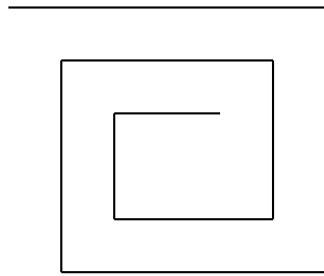
Écris des programmes pour dessiner les images suivantes. Tu peux choisir toi-même la position de départ de la tortue.



(a)

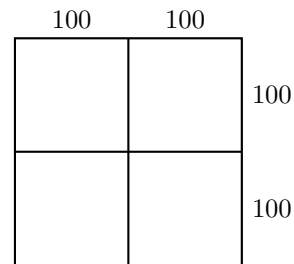


(b)



Tu peux choisir toi-même la longueur des traits.

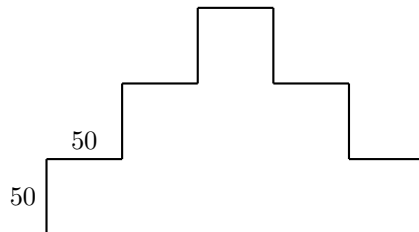
(c)



(d)

Exercice 4

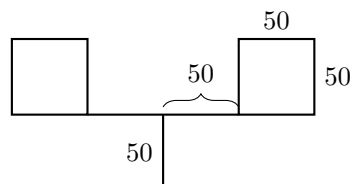
Écris un programme qui dessine l'image suivante:



Arrives-tu à modifier ton programme de sorte à n'utiliser que les instructions **fd 50** et **rt 90**?

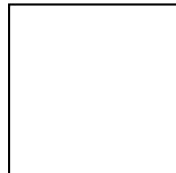
Exercice 5

Anna aimerait dessiner l'image suivante. Est-ce que tu peux l'aider?



2 L'instruction **repeat**

Si nous voulons dessiner un carré ayant une longueur de côté de 100,



nous pouvons utiliser le programme suivant:

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

Nous constatons que les deux instructions

```
fd 100  
rt 90
```

sont répétées quatre fois. Ne serait-il pas plus facile de dire à l'ordinateur de répéter lui-même ces instructions quatre fois à la place de les écrire quatre fois à la suite?

C'est justement ce que fait le programme suivant:

repeat	4	[fd 100 rt 90]
Instruction pour répéter un programme	Nombre de répétitions	Série d'instruc- tions à répéter

Exercice 6

Recopie le programme suivant et exécute-le:

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

Que dessine le programme? Peux-tu utiliser l'instruction **repeat** pour rendre le programme plus court?

Exercice 7

Écris le programme suivant pour voir ce qu'il dessine:

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

Écris une version plus courte de ce programme en utilisant **repeat**.

Exercice 8

Utilise l'instruction **repeat** pour écrire un programme qui dessine un carré avec une longueur de côté de 200.

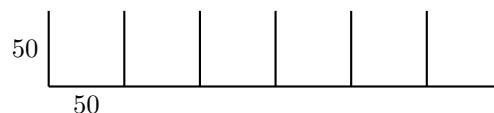
Exercice 9

Recopie le programme suivant et exécute-le:

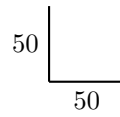
```
fd 100 rt 120
fd 100 rt 120
fd 100 rt 120
```

Que dessine le programme? Peux-tu utiliser l'instruction **repeat** pour rendre le programme plus court?

Nous voulons maintenant dessiner l'image suivante en utilisant l'instruction **repeat**:



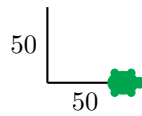
Avant de commencer à dessiner, nous devons trouver quel est le motif qui se répète. Nous pouvons par exemple choisir le motif suivant:



Si nous débutons en bas à gauche, le motif peut être dessiné par le programme suivant:

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

Après l'exécution de ce programme, la tortue est située comme dans l'image suivante et elle regarde vers la droite:



La tortue est déjà au bon endroit pour dessiner le prochain motif. Il ne reste plus qu'à la tourner pour qu'elle regarde vers le haut, ce que nous obtenons avec l'instruction `lt 90`.

Nous pouvons exécuter le programme pour vérifier qu'il est correct:

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

Nous obtenons la situation souhaitée:



Si nous exécutons à nouveau le programme, nous obtenons:



Nous voyons donc que notre idée fonctionne et nous pouvons répéter notre programme 6 fois:

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

Motif
Réorientation

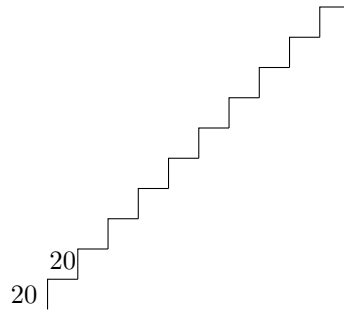
Nous pouvons résoudre beaucoup d'exercices en utilisant cette approche. Tous les exercices ci-dessous fonctionnent de cette façon. Il faut d'abord trouver le motif qui se répète. Il faut ensuite écrire deux programmes: un programme pour dessiner le *motif*, et ensuite en autre programme pour *réorienter* la tortue afin qu'elle soit prête à dessiner le motif suivant. Ton programme va avoir la forme suivante:

repeat *Nombre* [*Motif Réorientation*]

Exercice 10

Dessiner un escalier.

(a) Dessine un escalier avec dix marches de taille 20.



- Trouve le motif qui se répète et écris un programme pour le dessiner.
- Réfléchis comment réorienter la tortue afin qu'elle regarde dans la bonne direction pour la répétition prochaine du motif.
- Combine les deux programmes correctement pour résoudre l'exercice.

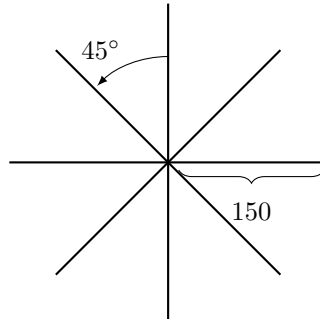
(b) Dessine un escalier avec 5 marches de taille 50.

(c) Dessine un escalier avec 20 marches de taille 10.

Exercice 11

Nous voulons maintenant dessiner des étoiles.

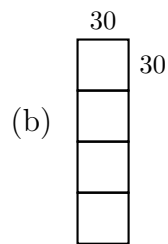
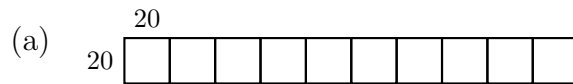
(a) Dessine l'étoile suivante:



(b) L'étoile ci-dessus est composée de 8 rayons de longueur 150. Peux-tu dessiner une étoile avec 16 rayons de longueur 100?

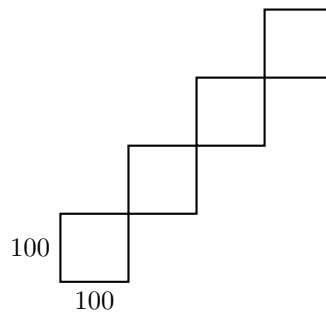
Exercice 12

Dessine les images suivantes:



Exercice 13

Écris un programme pour dessiner l'image suivante:



Exercice 14

Recopie le programme suivant et exécute-le:

```
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
```

Que dessine le programme? Peux-tu écrire une version plus courte de ce programme?

Mode déplacement

Normalement, la tortue se trouve en **mode crayon**. Ceci signifie qu'elle a un crayon dans la main et qu'elle dessine toujours en même temps qu'elle se déplace.

En **mode déplacement**, la tortue se déplace sur l'écran sans dessiner. Tu peux passer en mode déplacement avec l'instruction

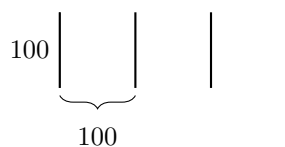
penup ou simplement **pu**.

Pour revenir en mode crayon, il faut utiliser l'instruction

pendown ou simplement **pd**.

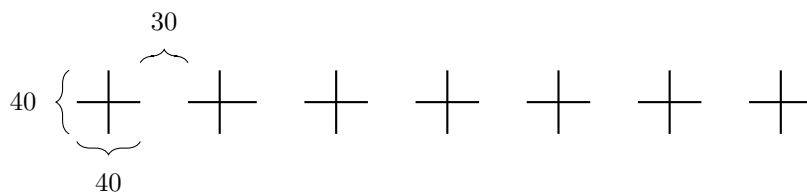
Exercice 15

Écris un programme pour dessiner l'image suivante:



Exercice 16

Écris un programme pour dessiner l'image suivante:



3 Nommer et appeler des programmes

Nous pouvons donner des noms aux programmes que nous écrivons. Si, par la suite, nous écrivons le nom d'un programme dans la ligne de commande, son contenu sera exécuté.

Le programme pour dessiner un carré avec une longueur de côté de 100 est:

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

Nous pouvons nommer ce programme **CARRE100**:

```
to CARRE100
repeat 4 [fd 100 rt 90]
end
```

Nous avons donc écrit deux fois le même programme, une fois sans nom et une fois avec nom.

Les programmes ayant un nom doivent être écrits dans l'**éditeur**. Dans ce manuel, ces programmes sont marqués par des boîtes grises. Quand nous avons terminé un programme dans l'éditeur, il faut cliquer sur le bouton avec la tortue pour fermer l'éditeur.

Le nom du programme peut être choisi librement; nous avons choisi **CARRE100** parce que nous voulions souligner qu'il s'agit de dessiner un carré avec une longueur de côté de 100. Les seules conditions pour le choix d'un nom sont qu'il n'est composé que de lettres et de nombres et qu'il puisse être écrit d'un seul trait, c'est à dire sans espace.

Rien n'a encore été dessiné sur l'écran car nous n'avons fait que donner un nom au programme et ne l'avons pas encore exécuté. Si nous écrivons maintenant le nom

CARRE100

dans la ligne de commande, le programme **repeat 4 [fd 100 rt 90]** sera exécuté. L'image suivante apparaît sur l'écran:



Observons à nouveau l'Exercice 12(a). Nous pouvons résoudre cet exercice plus simplement si nous écrivons d'abord un programme pour le motif à répéter, c'est-à-dire le carré avec une longueur de côté de 20, et lui donnons un nom:

```
to CARRE20
repeat 4 [fd 20 rt 90]
end
```

Après avoir dessiné **CARRE20**, la tortue est située en bas à gauche du carré:



Pour dessiner le carré suivant, elle doit se trouver en bas à droite, c'est pourquoi nous écrivons le programme

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

pour repositionner la tortue. Nous allons aussi donner un nom à ce programme:

```
to REPOSITIONNER20
rt 90 fd 20 lt 90
end
```

Nous pouvons donc résoudre l'Exercice 12(a) à l'aide de ces deux programmes comme suit:

```
repeat 10 [CARRE20 REPOSITIONNER20]
```

Le programme ci-dessus peut aussi avoir un nom. Par exemple:

```
to SUITE10
repeat 10 [CARRE20 REPOSITIONNER20]
end
```

Dans ce cas, nous disons que les programmes **CARRE20** et **REPOSITIONNER20** sont des **sous-programmes** du programme **SUITE10**.

Exercice 17

Écris un programme pour résoudre l'Exercice 12(b), en utilisant un programme pour dessiner des carrés avec une longueur de côté de 30. Le programme doit ressembler à ceci:

```
repeat 4 [CARRE30 REPOSITIONNER30]
```

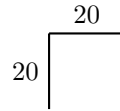
Tu dois donc écrire les programmes **CARRE30** et **REPOSITIONNER30** appropriés.

Exercice 18

Utilise le programme **CARRE100** comme sous-programme pour dessiner l'image de l'Exercice 13.

Exercice 19

Écris un programme pour dessiner la marche d'escalier



et utilise-le comme sous-programme pour résoudre l'Exercice 10(a).

Exercice 20

Résous à nouveau l'Exercice 11(a) en utilisant le sous-programme suivant:

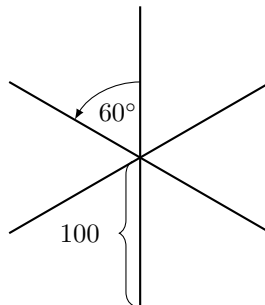
```
to LIGNE  
fd 150 bk 150  
end
```

Exercice 21

Écris un programme suivant:

```
to RAYON  
fd 100 bk 200 fd 100  
end
```

Utilise-le comme sous-programme du programme **ETOILE6** qui dessine l'image suivante:



Exercice 22

Résous à nouveau l'Exercice 15 et l'Exercice 16 à l'aide de sous-programmes.

Exercice 23

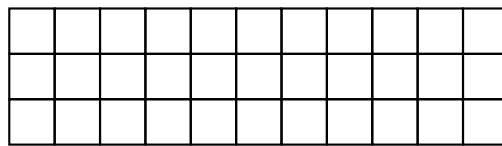
Ci-dessus, nous avons créé le programme **SUITE10**. Que fait le programme suivant?

```
SUITE10 fd 20 lt 90 fd 200 rt 90
```

Vérifie ta solution sur l'ordinateur.

Exercice 24

Écris un programme pour dessiner l'image suivante:



Exercice 25

Dessiner des carrés de tailles différentes.

- Écris un programme qui dessine un carré avec une longueur de côté de 50 et nomme-le **CARRE50**. Exécute-le pour vérifier qu'il est correct.
- Écris un programme qui dessine un carré avec une longueur de côté de 75.
- Exécute le programme

```
CARRE50  
CARRE75  
CARRE100
```

Qu'est-ce qui apparaît?

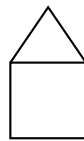
- Que changerais-tu dans le programme pour ajouter trois carrés plus grands?

Construire des maisons

Nous aimerions maintenant aider un architecte lors de la construction d'un quartier de maisons. Pour faciliter la construction, il a décidé de construire des maisons identiques. Nous lui faisons la proposition suivante:

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

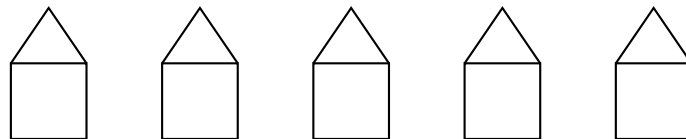
Ce programme dessine la maison ci-dessous:



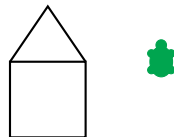
Exercice 26

Quelle est la position initiale de la tortue lorsqu'elle dessine cette maison? Trouve le chemin parcouru par la tortue lorsque le programme **MAISON** est exécuté. Où se trouve la tortue après avoir dessiné la maison? Dessine l'image correspondante et décris, comme dans l'Exercice 1, ce que fait chaque instruction.

La première maison a été construite et l'architecte constate que tout a bien fonctionné. Il aimerait donc utiliser ce programme comme motif pour construire une première rue de maisons. À la fin, la rue devra ressembler à ceci:



Comme l'architecte construit toutes les maisons de la même manière, il peut utiliser le motif **MAISON** cinq fois et ne doit donc pas réfléchir à chaque fois comment construire la maison. Il dit à la tortue de construire la première maison à gauche et la fait ensuite aller au point de départ de la deuxième maison:



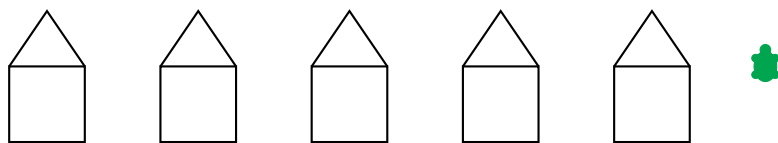
Afin d'y arriver, il utilise le programme suivant:

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

La tortue peut maintenant dessiner une nouvelle maison à cet endroit et ensuite aller au point de départ de la maison suivante. C'est ce qu'elle fait jusqu'à ce que les cinq maisons soient dessinées. Nous devons donc répéter le programme ci-dessus cinq fois pour dessiner une suite de cinq maisons identiques. Nous nommons le programme correspondant **RANGEEMAISONS**:

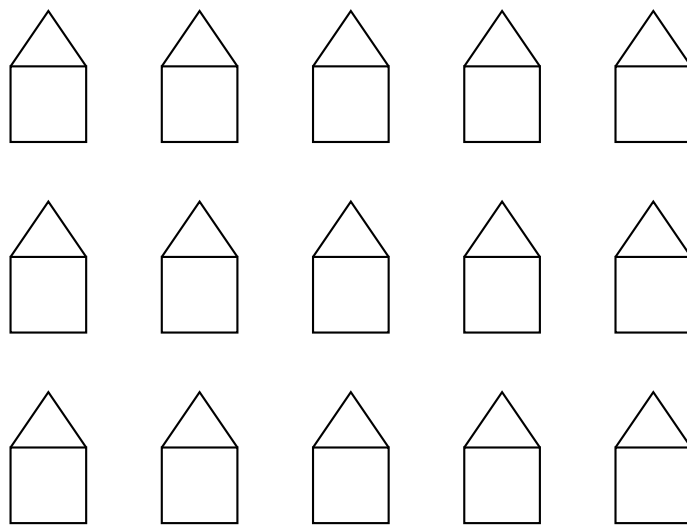
```
to RANGEEMAISONS  
repeat 5 [MAISON rt 90 pu fd 50 lt 90 pd]  
end
```

La tortue se trouve maintenant tout à droite, à l'endroit où la maison suivante serait dessinée:



Exercice 27

Nous voulons maintenant ajouter quelques nouvelles rues au quartier. Utilise le programme **RANGEEMAISONS** comme base pour dessiner l'image suivante:



Indice: Après chaque rangée de maisons, la tortue doit se rendre au bon endroit pour commencer la rangée suivante.

Lignes épaisses et carrés noirs

Exercice 28

Dessiner des lignes épaisses avec le programme **EPAIS**

Donne le nom **EPAIS** au programme

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

et écris ensuite

```
EPAIS
```

Que dessine la tortue? Dessine sur une feuille comment l'image a été produite avec ton crayon.

Exercice 29

Répète le programme **EPAIS** 100 fois avec le programme

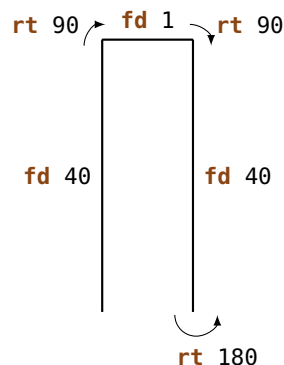
```
repeat 100[EPAIS]
```

Qu'est-ce qui apparaît sur l'écran?

Exercice 30

Dans cet exercice, nous allons dessiner des lignes épaisses et des surfaces noires. Nous avons vu dans l'Exercice 28 qu'une ligne épaisse peut être dessinée comme ceci:

```
to EPAIS40
fd 40
rt 90
fd 1
rt 90
fd 40
rt 180
end
```



La ligne épaisse est créée en dessinant deux lignes si près l'une de l'autre qu'on ne voit plus qu'une ligne qui est plus épaisse. Recopie et exécute le programme **EPAIS40**.

Exercice 31

Nous pouvons considérer une ligne épaisse de longueur 40 comme un rectangle de largeur 1 et longueur 40. Après avoir dessiné `EPAIS40`, la tortue se situe en bas de la deuxième ligne et regarde vers le haut. Si nous répétons le programme `EPAIS40`, la tortue passera à nouveau sur cette deuxième ligne, et nous aurons un rectangle de largeur 2 et longueur 40. Chaque répétition ajoute donc une nouvelle ligne. Si nous répétons `EPAIS40` 40 fois, nous obtiendrons le carré noir avec une longueur de côté de 40. Vérifie que ça fonctionne en répétant `EPAIS40` 40 fois.

Écris un programme nommé `NOIR40` pour dessiner un carré noir avec une longueur de côté de 40.

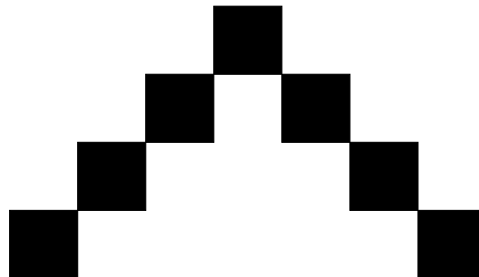
Exercice 32

Utilise le programme `NOIR40` pour dessiner l'image suivante:



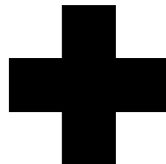
Exercice 33

Utilise le programme `NOIR40` pour dessiner l'image suivante:



Exercice 34

Dessine l'image suivante:



Exercice 35

Écris un programme pour dessiner l'image suivante:



Exercice 36

L'architecte décide de commander le toit chez un autre fournisseur. Il reçoit donc deux motifs: un motif **TOIT** et un motif **PARTIEINFERIEURE**. Écris deux programmes pour dessiner ces motifs et assemble-les dans un nouveau programme **MAISON1** afin de dessiner une maison.

Exercice 37

Les maisons de l'Exercice 27 sont très simples à construire. Sois créatif: conçois une nouvelle maison pour construire un nouveau quartier.

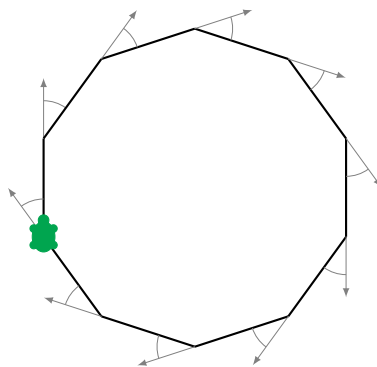
4 Polygones réguliers et cercles

Polygones réguliers

Un polygone régulier possède n sommets et n côtés de même longueur. Par exemple, pour dessiner un décagone (un polygone régulier à 10 côtés) avec un crayon, tu dois dessiner 10 lignes en changeant « un peu » de direction, c'est-à-dire en tournant un peu, après chaque ligne.

De combien faut-il chaque fois tourner?

Pour dessiner un polygone régulier, on tourne et bouge la tortue plusieurs fois. À la fin, elle retrouve la position initiale et elle regarde dans la même direction qu'au debout.



Cela signifie qu'on a tourné d'un total de 360° . Par conséquent, lorsqu'on dessine un décagone régulier, on tourne dix fois d'un angle identique. Cet angle est:

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

Il faut donc à chaque fois tourner de 36° : **rt 36**. Essayons ceci avec le programme suivant:

```
repeat 10 [ fd 50           rt 36 ]
            Longueur du côté   Rotation de 36°
```

Exercice 38

Dessine les polygones réguliers suivants:

- (a) un polygone régulier à 5 côtés de longueur 180 (un pentagone) ,
- (b) un polygone régulier à 12 côtés de longueur 50 (un dodécagone) ,
- (c) un polygone régulier à 4 côtés de longueur 200 (un carré) ,
- (d) un polygone régulier à 6 côtés de longueur 100 (un hexagone) ,
- (e) un polygone régulier à 3 côtés de longueur 200 (un triangle) et
- (f) un polygone régulier à 18 côtés de longueur 20 .

Si nous voulons dessiner un polygone régulier à 7 côtés (un heptagone), nous avons un problème: 360 n'est pas divisible sans reste par 7. Dans ce cas, nous pouvons laisser l'ordinateur calculer le résultat en écrivant

```
360/7
```

(« / » signifie « diviser » pour l'ordinateur). L'ordinateur calcule le résultat exact et nous pouvons dessiner l'heptagone avec une longueur de côté de 100 comme suit:

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

Vérifie que ce programme fonctionne.

Dessiner des cercles

On ne peut pas dessiner de cercles parfaits avec les instructions **fd** et **rt**. Mais comme tu l'as certainement déjà remarqué, des polygones réguliers comportant de nombreux sommets ressemblent beaucoup à des cercles. Nous pouvons donc obtenir des cercles en dessinant des polygones réguliers ayant un grand nombre de sommets ainsi que des côtés de petite longueur.

Exercice 39

Teste les programmes suivants:

```
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 signifie 3 pas et demi.

Exercice 40

- (a) Comment ferais-tu pour dessiner un tout petit cercle? Écris un programme pour le faire.
- (b) Comment dessinerais-tu un grand cercle? Écris un programme pour le faire.

Exercice 41

Essaie de dessiner les demi-cercles suivants. Tu peux choisir toi-même leur taille.



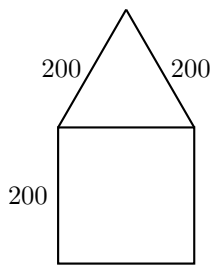
(a)



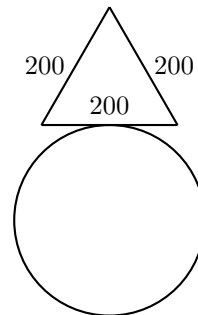
(b)

Exercice 42

Utilise tes nouvelles connaissances pour dessiner les images suivantes. Tu peux choisir toi-même la taille du cercle.



(a)



(b)

Motifs créatifs

Dessine un heptagone (un polygone régulier à 7 côtés:

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

Tourne ensuite la tortue de 10° avec

```
rt 10
```

Répète plusieurs fois ces deux programmes et observe le résultat. Après chaque heptagone, la tortue tourne de 10° avec **rt 10**. Si nous voulons qu'elle retourne à sa position initiale, nous devrons répéter cette opération

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

fois. Observons ce que dessine le programme suivant:

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

Exercice 43

Dessine un dodécagone (un polygone régulier à 12 côtés) avec une longueur de côté de 70 et tourne-le 18 fois pour retourner à la position de départ.

Indice: Tu peux commencer par écrire un programme pour un dodécagone avec une longueur de côté de 70 et le nommer **DODECAGONE**. Ensuite, il ne restera qu'à compléter le programme


















```
repeat 18 [DODECAGONE rt ... ]
```

Exercice 44

Réfléchis à un exercice semblable à l'Exercice 43 et écris un programme pour le résoudre.

Couleurs

Pour améliorer nos motifs créatifs, nous pouvons aussi leur ajouter des couleurs. En effet, la tortue peut dessiner avec la couleur de son choix. À chaque couleur est associée un numéro. Les couleurs disponibles sont les suivantes:

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

Avec l'instruction

setpencolor	X
Instruction pour changer la couleur	Numéro de la couleur souhaitée

la tortue passe à la couleur numéro **X**. Nous pouvons abrégé l'instruction par **setpc**.

De cette façon, nous pouvons dessiner des motifs amusants, comme par exemple celui créé par le programme suivant. Nous commençons par nommer deux programmes qui dessinent des cercles de tailles différentes:

```
to CERCLE3
repeat 360 [fd 3 rt 1]
end

to CERCLE1
repeat 360 [fd 1 rt 1]
end
```

Nous utilisons maintenant ces cercles pour dessiner des motifs semblables aux précédents:

```
to MOTIF3
repeat 36 [CERCLE3 rt 10]
end

to MOTIF1
repeat 18 [CERCLE1 rt 20]
end
```

Nous essayons maintenant d'utiliser des couleurs:

```
setpc 2
MOTIF3 rt 2
setpc 3
MOTIF3 rt 2
```

```
setpc 4
MOTIF3 rt 2
setpc 5
MOTIF3 rt 2
```

```
setpc 6
MOTIF1 rt 2
setpc 15
MOTIF1 rt 2
```

```
setpc 8
MOTIF1 rt 2
setpc 9
MOTIF1 rt 2
```

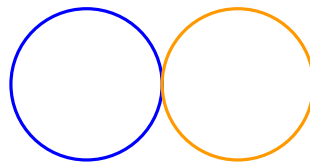
Tu peux volontiers continuer le travail et dessiner davantage. Tu peux également dessiner un motif selon ton imagination.

Exercice 45

Dessine **MOTIF3** en orange. Utilise ensuite l'instruction **setpc 7** pour passer en blanc. Qu'est-ce qui se passe si tu exécutes à nouveau **MOTIF3**?

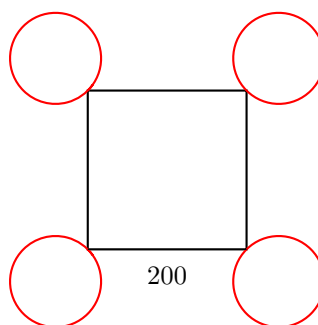
Exercice 46

Dessine l'image suivante. Au début la tortue se trouve à l'intersection des deux cercles.



Exercice 47

Écris un programme pour dessiner l'image suivante. Tu peux choisir toi-même la taille des cercles.



5 Programmes avec paramètres

Dans la Leçon 3, nous avons appris à donner des noms aux programmes et à les appeler par ces noms pour que l'ordinateur dessine les images souhaitées. Dans la Leçon 4, nous avons appris à dessiner des polygones. Cependant, il est embêtant de devoir écrire un nouveau programme pour chaque polygone ayant un autre nombre de sommets.

Considérons par exemple les trois programmes suivants:

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

Les programmes sont très semblables et les seules différences sont les nombres jaunes 7, 12 et 18. Ces nombres déterminent le nombre de sommets. Nous voulons maintenant écrire un programme avec lequel nous pourrions dessiner n'importe quel polygone:

```
to POLYGONE :SOMMETS
repeat :SOMMETS [fd 50 rt 360/:SOMMETS]
end
```

Qu'avons-nous fait? Partout dans le programme, nous avons remplacé le nombre de sommets par un nom, dans ce cas **:SOMMETS**. Afin que l'ordinateur sache dès le départ que nous voulons choisir librement le nombre de sommets plus tard, nous devons aussi écrire **SOMMETS** précédé de **:** après le nom du programme.

Si nous écrivons maintenant l'instruction **POLYGONE 12** dans la ligne de commande, l'ordinateur remplace **:SOMMETS** par le nombre 12 partout dans le programme

```
repeat :SOMMETS [fd 50 rt 360/:SOMMETS]
```

12 12

Essaie par toi-même:

```
POLYGONE 3
POLYGONE 4
POLYGONE 5
POLYGONE 6
```

Nous appelons **:SOMMETS** un **paramètre**. Dans l'exemple ci-dessus, 3, 4, 5 et 6 sont les **valeurs du paramètre :SOMMETS**. L'ordinateur sait qu'il s'agit d'un paramètre grâce au **:**. C'est pour ceci qu'il faut écrire un **:** avant chaque occurrence du paramètre.

Exercice 48

Les programmes suivants dessinent des carrés avec des longueurs de côtés différentes:

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

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

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

Nous pouvons considérer les nombres jaunes 100, 50 et 200 comme valeurs d'un paramètre qui détermine la longueur des côtés du carré. Écris un programme qui peut dessiner un carré de taille quelconque en utilisant un paramètre **:LONGUEUR**:

```
to CARRE :LONGUEUR  
...  
end
```

Exercice 49

Les programmes suivants dessinent des cercles de tailles différentes:

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

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

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

Écris un programme pour dessiner des cercles de tailles quelconques en utilisant un paramètre. Essaie-le avec les valeurs de paramètre 1, 2, 3, 4 et 5. Tu peux choisir toi-même le nom du programme et le nom du paramètre. Tu dois juste faire attention à ne pas oublier le double-point avant le paramètre.

Exercice 50

Te rappelles-tu comment dessiner des lignes épaisses (Exercice 28)? Écris un programme pour dessiner une ligne épaisse de longueur quelconque en utilisant un paramètre.

Indice: Tu peux commencer par écrire un programme pour une ligne épaisse de longueur 100 et un programme pour une ligne épaisse de longueur 50 afin de trouver où il faut utiliser le paramètre.

Exercice 51

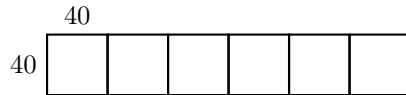
Écris un programme pour dessiner un triangle équilatéral de taille quelconque en utilisant un paramètre. En utilisant ce programme, dessine ensuite l'un après l'autre les triangles ayant les grandeurs suivantes:

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

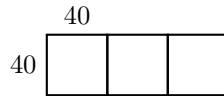
Qu'est-ce qui apparaît?

Exercice 52

Maintenant nous aimerions bien de dessiner des carrés l'un à côté de l'autre. La taille de chaque carré est 40. Écris un programme **CARRES** avec un paramètre **:NOMBRE**. Le paramètre **:NOMBRE** détermine combien de carrés la tortue dessinera. Ainsi, lorsque l'on appelle **CARRES 6**, l'image suivante apparaît sur l'écran:

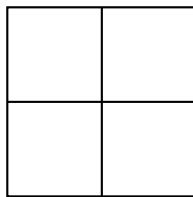


Voici l'image après l'exécution de **CARRES 3** :



Exercice 53

Écris un programme pour dessiner l'image suivante qui est composée de 4 carrés. Utilise un paramètre pour choisir la taille d'un carré librement.

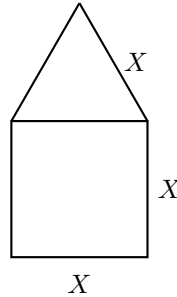


Exercice 54

Écris un programme qui dessine des hexagones réguliers de tailles quelconques en utilisant un paramètre. Essaie ton programme en dessinant des hexagones ayant les longueurs de côtés 40, 60 et 80.

Exercice 55

Écris un programme pour dessiner des maisons de tailles quelconques en utilisant un paramètre `:X` selon l'image suivante.



Programmes à plusieurs paramètres

Un programme peut avoir plus d'un paramètre. Lorsque nous dessinons des polygones, nous pouvons définir un paramètre `:SOMMETS` pour le nombre de sommets et un paramètre `:LC` pour la longueur des côtés.

Dans les programmes suivants, le paramètre `:SOMMETS` est marqué en jaune et le paramètre `:LC` en vert :

```
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]
```

Nous pouvons maintenant écrire un programme pour dessiner les polygones de notre choix :

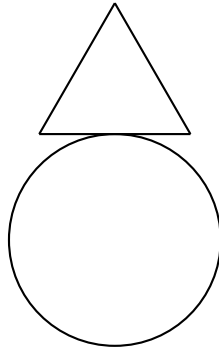
```
to PGONE :SOMMETS :LC
repeat :SOMMETS [fd :LC rt 360/:SOMMETS]
end
```

Teste le programme `PGONE` avec les instructions suivantes :

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


Exercice 56

Écris un programme avec deux paramètres pour dessiner l'image suivante. La taille du cercle et la taille du triangle doivent pouvoir être choisies librement.



Exercice 57

Le programme

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

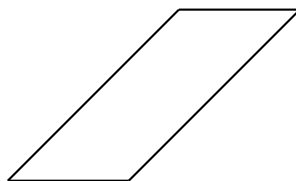
dessine un rectangle de largeur 100 et longueur 200. Vérifie qu'il fonctionne et écris un programme avec deux paramètres pour dessiner des rectangles de largeur et longueur quelconques.

Exercice 58

Le programme

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

dessine un parallélogramme :



Écris un programme qui peut dessiner de tels parallélogrammes avec des longueurs de côtés quelconques en utilisant deux paramètres.

Exercice 59

Dessine une fleur. Pour cela, dessine un cercle avec

```
PGONE 360 2
```

puis tourne un peu la tortue avec

```
rt 20
```

Dessine ensuite un nouveau cercle avec

```
PGONE 360 2
```

et continue ainsi avec

```
rt 20 PGONE 360 2 rt 20 PGONE 360 2 ...
```

Lorsque la fleur est terminée, la tortue se trouve de nouveau à sa position de départ. La tortue a dessiné 18 cercles, chacun suivi d'une rotation de 20° . La tortue a donc tourné d'un total de $18 \times 20^\circ = 360^\circ$.

En résumé, nous avons le programme suivant :

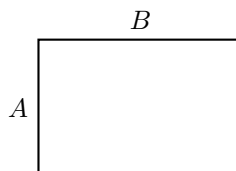
```
repeat 18 [PGONE 360 2 rt 20]
```

Essaie-le.

- Il est aussi possible de dessiner des fleurs avec, par exemple, 10 ou 20 pétales (cercles). Comment le ferais-tu? Écris un programme et essaie-le.
- Peux-tu écrire un programme pour dessiner des fleurs avec un nombre quelconque de pétales (cercles) en utilisant un paramètre?
- Parviens-tu à écrire un programme dans lequel tu peux choisir librement :
 - le nombre de pétales (cercles) et
 - la taille des cercles?

Exercice 60

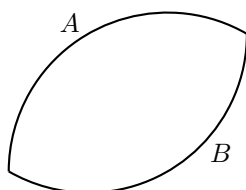
Écris un programme pour dessiner des rectangles quelconques avec des couleurs quelconques :



Tu dois donc pouvoir choisir librement les longueurs des côtés A et B ainsi que la couleur du rectangle.

6 Comment dessiner des fleurs et comment passer des paramètres à des sous-programmes

Nous pouvons considérer une feuille



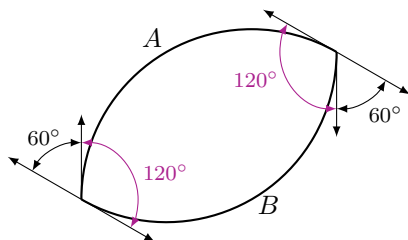
comme un assemblage de deux arcs de cercle A et B . Pour dessiner un arc de cercle, nous pouvons par exemple utiliser le programme suivant :

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

Vérifie comment ce programme fonctionne.

Ce programme ressemble beaucoup à celui que nous avons utilisé pour dessiner un cercle. À la place de 360 petits pas suivis de rotations de 1° , on ne bouge que 120 [fd 3 rt 1] pour dessiner un tiers du cercle ($\frac{360^\circ}{3} = 120^\circ$).

La question est maintenant de combien la tortue doit tourner sur place avant de dessiner l'arc de cercle B pour la partie inférieure de la feuille. Pour ceci, observons l'image suivante :



Si nous voulons que la position finale soit la même que la position de départ, nous devons tourner la tortue d'un total de 360° . Dans la partie A , nous la tournons de 120° et dans la partie B également de 120° . Il reste donc

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

que l'on doit répartir de manière égale entre les deux rotations aux pointes de la feuille :

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

Nous obtenons le programme suivant :

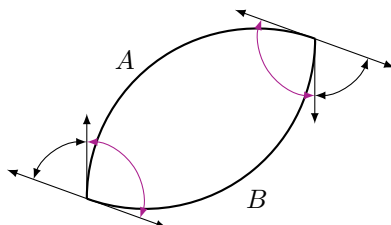
```
repeat 120 [fd 2 rt 1]
rt 60
repeat 120 [fd 2 rt 1]
rt 60
```

ou plus simplement :

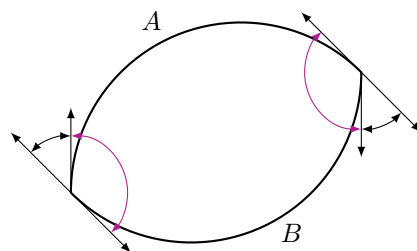
```
repeat 2 [repeat 120 [fd 2 rt 1] rt 60]
```

Vérifie comment le programme fonctionne.

Il est aussi possible de dessiner des feuilles plus étroites (les arcs *A* et *B* sont plus courts) ou plus grosses (les arcs *A* et *B* sont plus longs).



ANGLE=120°



ANGLE=135°

Pour ceci, nous utilisons à nouveau un paramètre, que nous pouvons par exemple nommer **:ANGLE**. Nous calculons ensuite la rotation aux pointes de la feuille comme suit :

Avant de dessiner la partie *B* de la feuille, il faut faire la moitié d'un tour entier, c'est-à-dire $\frac{360^\circ}{2} = 180^\circ$. La rotation à la pointe de la feuille est donc de

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

Nous pouvons maintenant écrire le programme suivant dans l'éditeur :

```
to FEUILLE :ANGLE
repeat 2 [repeat :ANGLE [fd 2 rt 1] rt 180-:ANGLE]
end
```

Essaie ce programme en écrivant les instructions suivantes dans la ligne de commande :

```
FEUILLE 20  
FEUILLE 40  
FEUILLE 60  
FEUILLE 80  
FEUILLE 100
```

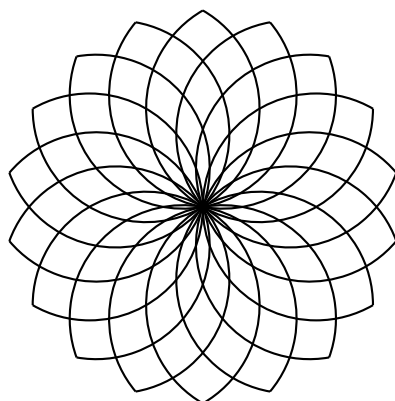
Que se passe-t-il?

Exercice 61

Motif créatif

Commence par dessiner une feuille avec le programme suivant :

```
FEUILLE 100  
rt 20  
FEUILLE 100  
rt 20  
FEUILLE 100  
....
```



Combien de fois dois-tu répéter les instructions **FEUILLE** et **rt 20** pour dessiner une fleur complète?

Écris le programme pour cette fleur en une seule ligne en utilisant une instruction **repeat** adéquate. (N'oublie pas que la somme des rotations **rt** effectuées après chaque feuille doit donner un total de 360°.)

Exercice 62

Dans le programme **FEUILLE**, l'instruction **fd 2** détermine la taille du cercle que nous utilisons pour l'arc de cercle de longueur **:ANGLE**. Nous pouvons remplacer la valeur 2 par un paramètre nommé **:LARG** (largeur). Écris un programme

```
FEUILLES :ANGLE :LARG
```

avec les paramètres **:ANGLE** et **:LARG**, dans lequel nous pouvons déterminer en même temps la longueur et la largeur des feuilles. Essaie-le avec les instructions suivantes :

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

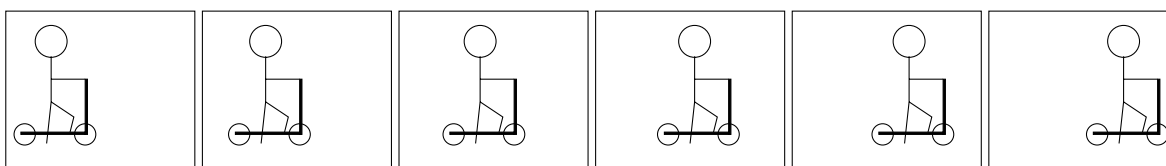
Tourne ensuite la tortue à droite de 80° et recommence le programme ci-dessus.

Exercice 63

Réfléchis à d'autres motifs créatifs.

7 Comment programmer les animations

Sais-tu comment les films d'animation sont faits? Cela fonctionne exactement de la même façon que dans le feuilleteuse. D'abord on dessine quelques images qui sont chacune seulement un peu différente l'une de l'autre. Dans l'exemple suivant le garçon sur la planche bouge toujours un peu vers la droite d'une image à une autre :



Quand on place les images l'une à l'autre et on les fait défiler rapidement avec un pouce, on a l'impression que le garçon bouge vraiment sur la planche de la gauche vers la droite. Images en mouvement sont appelés **Animations**.

Dans cette leçon nous allons apprendre comment nous pouvons programmer des animations à l'aide de la tortue.

Comment dessiner un carré qui laisse des traces

Pour notre première animation nous avons choisi un objet qui n'est pas trop compliqué et que nous connaissons déjà depuis longtemps : On fait bouger un carré de la gauche vers la droite.



On connaît déjà le programme suivant qui dessine un carré de taille 100 :

```
to CARRE100
repeat 4 [fd 100 rt 90]
end
```

Une fois le carré est dessiné on déplace la tortue un peu vers la droite et on dessine encore un autre carré. C'est ce qu'on va répéter plusieurs fois.

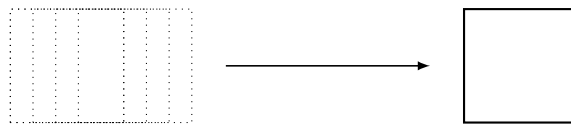
Dans le programme suivant on dessine le carré précédent 120 fois :

```
to CARREMOUVEMENT
repeat 120 [CARRE100 rt 90 fd 4 lt 90]
end
```

Exercice 64

Écris le programme **CARRE100** et **CARREMOUVEMENT** dans l'éditeur.
Essaye **CARREMOUVEMENT**. Qu'est-ce que apparaît sur l'écran?

Tu peux bien voir que les traces de *tous* les carrés sont dessinés. Mais pour notre animation nous n'aimerions voir que le dernier carré et nous voudrions effacer les traces antérieures.

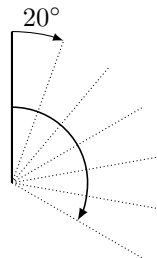


Exercice 65

Fais déplacer un carré du bas vers le haut au lieu de la gauche vers la droite.

Exercice 66

Écris un programme pour un segment de la longueur de 20. Utilise ce programme pour déplacer ce segment autour de son extrémité inférieure dans le sens des aiguilles d'une montre :



Comment dessiner un carré et comment l'effacer à nouveau

Pour brouiller les pistes nous devons apprendre à effacer les images que l'on vient de tracer. A cet effet la tortue doit utiliser une gomme au lieu d'un crayon. La tortue passe du **mode crayon** en **mode effacement** avec l'instruction **penerase** ou simplement **pe**.

Exercice 67

Réfléchis sur le fait ce que le programme `CARRE100 pe CARRE100` fait sans utilisation de l'ordinateur.

Pour faire la tortue dessiner à nouveau nous devons le lui annoncer. Pour y arriver, on utilise une nouvelle instruction : `penpaint` ou simplement `ppt`. On peut appliquer cette instruction directement dans notre programme de l'Exercice 67.

On obtient alors ce programme ci :

```
CARRE100 pe CARRE100 ppt
```

Exercice 68

Essaye le programme ci-dessus. Qu'est-ce qui se passe? Peux-tu expliquer pourquoi?

Comment faire un carré attendre un peu

Comme tu l'as sûrement déjà compris dans l'Exercice 68, le carré sera effacé immédiatement après son dessin. On ne remarque même pas qu'un carré a été dessiné. Avant que on efface un carré, on doit faire patienter la tortue un peu.

On peut le faire ainsi :

<code>wait</code>	4
L'instruction d'attente	Le temps d'attente

Exercice 69

Essaye le programme suivant :

```
CARRE100 wait 4 pe CARRE100 ppt
```

Comment faire pour déplacer un carré de la gauche vers la droite

Nous sommes maintenant prêt à inclure des instructions pour l'effacement d'un carré et pour l'attente dans notre programme **CARREMOUVEMENT** :

```
to CARREMOUVEMENT
repeat 120 [CARRE100 wait 4 pe CARRE100 rt 90 fd 4 lt 90 ppt]
end
```

Essaye-le. Si la tortue te dérange pendant l'animation, tu peux démarrer le programme avec l'instruction **hideturtle** (ou plus courte : **ht**), ce qui rend la tortue invisible. Tu vas te rendre compte immédiatement que l'animation devient plus rapide. Finis le programme par l'instruction **showturtle** (ou plus courte : **st**) juste avant **end**. Cette instruction rend la tortue à nouveau visible.

Exercice 70

Déplace un carré de taille 50×50 vers le haut.

Exercice 71

Modifie le programme **CARREMOUVEMENT** de telle manière que le carré se déplace deux fois plus vite vers la droite comme avant.

Exercice 72

Arrives-tu à modifier le programme **CARREMOUVEMENT** de telle manière que le carré se déplace deux fois plus lentement vers la droite?

Exercice 73

Modifie le programme **CARREMOUVEMENT** de telle manière que le carré se déplace de la droite vers la gauche au lieu de la gauche vers la droite.

Exercice 74

Réfléchis sur le fait ce que le programme suivant fait.

```
to CARREMOUVEMENT1
ht
repeat 50 [CARRE100 wait 5 pe CARRE100 fd 3 rt 90 fd 3 lt 90 ppt]
CARRE100
st
end
```

Puis vérifie ton hypothèse en exécutant le programme :

Exercice 75

Tout d'abord, réfléchis sur le fait ce que le programme suivant fait.

```
to CERCLES
ht
repeat 360 [CARRE100 wait 4 pe CARRE100 fd 5 rt 1 ppt]
CARRE100
st
end
```

Ensuite, vérifie si ton hypothèse était juste à l'aide de l'ordinateur.

Exercice 76

Modifie le programme **CERCLES** afin d'effectuer la rotation d'un carré quatre fois plus vite.

Exercice 77

Quel est l'effet du programme suivant?

```
repeat 6 [CERCLES]
```

Exercice 78

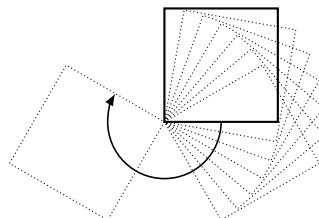
Écris le programme suivant

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

et utilise-le pour créer une animation, dans laquelle la Terre tourne autour du Soleil. Réfléchis de quelle façon tu vas programmer le Soleil.

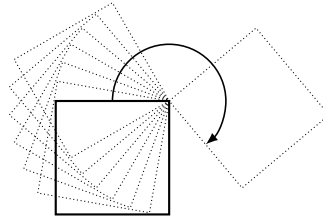
Exercice 79

Tourne un carré autour de son coin inférieur gauche dans le sens horaire. Tu peux choisir toi-même la taille de ce carré :



Exercice 80

Maintenant tourne un carré autour de son coin supérieur droit dans le sens horaire :



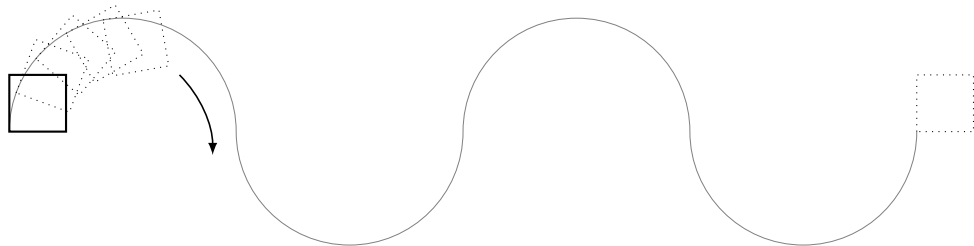
Si tu connais déjà la notion de **paramètres**, tu peux travailler sur les exercices suivantes.

Exercice 81

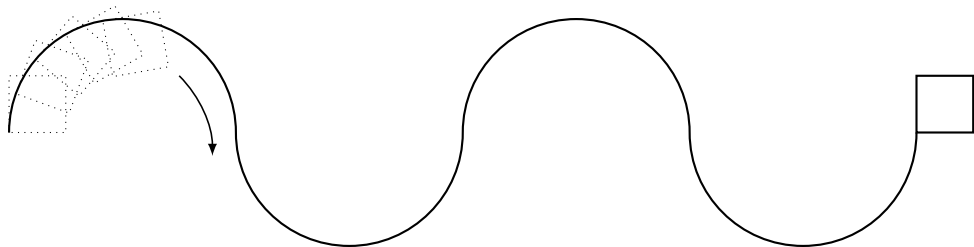
Écris un programme avec *deux paramètres* afin de déplacer un carré à partir de la gauche vers la droite. Un paramètre détermine la longueur de son côté, autre paramètre détermine la rapidité avec laquelle il se déplace.

Exercice 82

(a) Déplace un carré sur la voie tracée ci-dessous, qui se compose de 4 demi-cercles. La taille de ce carré doit pouvoir être choisie librement.



(b) Maintenant, on veut faire aussi la trace de ce déplacement comme un chemin.



(c) Arrives-tu à améliorer le programme en (b) de telle sorte que le nombre de demi-cercles sera donnée également par un paramètre?

Mes notes



Liste des instructions

- fd 100** prends 100 pas en avant
- bk 50** prends 50 pas en arrière
- cs** supprimer tout et recommencer
- rt 90** pivotes de 90 degrés vers la droite
- lt 90** pivotes de 90 degrés vers la gauche
- repeat 4 [...]** le programme dans [...] est quatre fois répété
 - pu** la tortue se met en mode déplacement
 - pd** la tortue passe au mode crayon
- setpc 3** change la couleur du crayon à la couleur 3
- to NAME** créera un programme avec un nom
- to NAME :PARAMETER** créera un programme avec un nom et un paramètre
 - end** tous les programmes avec un nom finissent par cette instruction
 - pe** la tortue se met en mode effacement
 - ppt** la tortue revient de mode effacement au mode crayon
- wait 5** fait la tortue attendre 5 unités de temps



Programmer avec LOGO

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

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