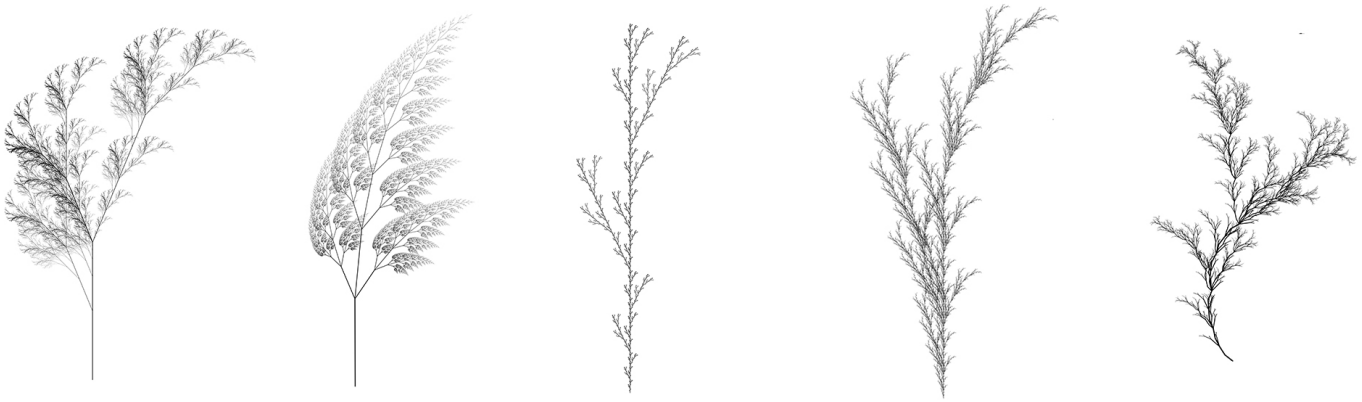
L-Systèmes

- Les systèmes de Lindenmayer, appelés aussi L-Systèmes, ont été imaginés par le biologiste Aristid Lindenmayer(1925-1989) et modélisent le processus de développement et de prolifération de plantes.
- Le concept central des L-Systèmes est la représentation d'une plante par une chaîne de caractères. Cela permet de modéliser son évolution, voire sa destruction par des agents pathogènes, au moyen de règles de transformations de ces caractères.
- On utilisera ici le module `turtle` pour représenter un L-Système.
- Cette activité permet de travailler les fonctions et la notion de pile.

In [4]:
```
from turtle import *
```



1. Un alphabet pour coder une figure.

Définissons une figure f par la donnée d'un triplet contenant :

- Une longueur L.
- Un pas de rotation a, donné en degrés.
- Un mot, appelé motif, utilsant les caractères de l'ensemble $\{F; +; -\}$ avec comme convention :
  - F : avancer de L
  - $+$: tourner à gauche de a degrés
  - $-$: tourner à droite de a degrés

Par exemple, la figure $f(50; 90; F + F + F + F)$ représente un carré dé 50 unités de côté.

Exercice 1 :

1. Que représente la figure $f(20; 60; F + +F + +F)$?

Réponse : Un triangle équilatéral de côté 20 unités.

2. Ecrire la suite d'instructions turtle qui dessine cette figure.

In [3]:
```
forward(20)
left(60)
left(60)
forward(20)
left(60)
left(60)
forward(20)

mainloop()
```

3. A quelle figure f correspond le dessin ci-dessous ?

Réponse : $f(20, 60, F + F - -F + F)$

4. Ecrire la suite d'instructions turtle qui dessine cette figure.

In [5]:
```
forward(20)
left(60)
forward(20)
right(60)
right(60)
forward(20)
left(60)
forward(20)
mainloop()
```

Exercice 2 :

Ecrire la fonction `dessiner(unite, angle, motif)` qui :

- reçoit en paramètres :
  - `unite` : un nombre représentant la longueur L.
  - `angle` : un nombre représentant l'angle a de rotation.
  - `motif` : le motif m de la figure sous forme d'une chaîne de caractères
- affiche le dessin de la figure $f(L, a, m)$ avec le module `turtle`

Par exemple, `dessiner(50,60,'F++F++F')` dessine un triangle équilatéral

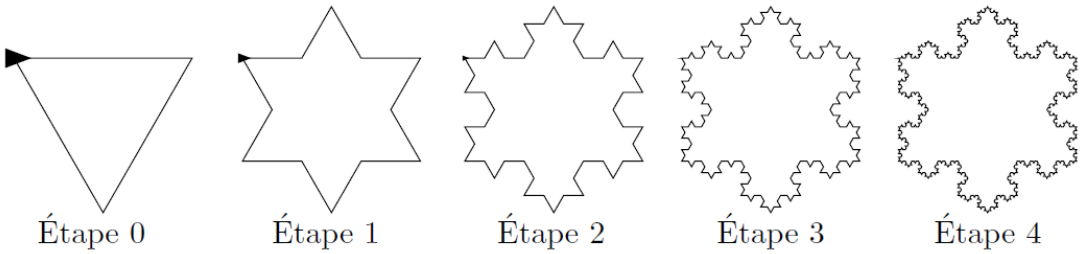
In [5]:
```
from turtle import *

def dessiner(unite, angle, motif) :
    for car in motif:
        if car=='F':
            forward(unite)
        elif car=='+':
            left(angle)
        elif car=='-':
            right(angle)
```

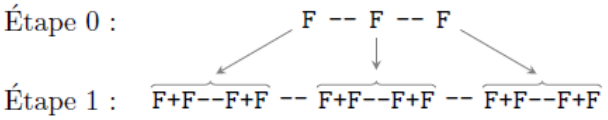
In [8]:
```
#triangle
dessiner(50,60,'F++F++F')
mainloop()
```

2. Premiers L-Systèmes

L'intérêt des L-Systèmes est de permettre de décrire simplement l'évolution d'une figure. Prenons l'exemple du flocon de Von Koch (l'échelle d'une image à l'autre a été ajustée pour une meilleure visibilité) :



En choisissant un pas d'angle de rotation de 60°, les motifs des deux premières figures sont :



A chaque étape, chaque lettre F représentant un segment est remplacé par le motif $F+F--F+F$. Un L-système est la donnée d'un *axiome* (motif de départ) et d'une *règle* ou d'un ensemble de règles.

Dans notre exemple, l'axiome est $F--F--F$ et la règle $F+F--F+F$.

Exercice 3 :

Compléter la fonction `suivant(motif,regle)` qui :

- reçoit en paramètres deux chaînes de caractères :
  - motif : le motif de la figure à une étape donnée
  - regle : la regle d'évolution de la figure
- renvoie une chaîne de caractères représentant la figure à l'étape suivante.

Par exemple : `suivant('F--F--F','F+F--F+F')` renvoie `'F+F--F+F--F+F--F+F--F+F--F+F--F+F'`

In [ ]:
```
def suivant(motif,regle):
    res=...
    for ... in motif:
        if car=='F':
            res=...
        else:
            res=...
    return res
```

In [7]:
```
def suivant(motif,regle):
    res=''
    for car in motif:
        if car=='F':
            res=res+regle
        else:
            res=res+car
    return res
```

In [8]:
```
#test
assert suivant('F--F--F','F+F--F+F')=='F+F--F+F--F+F--F+F--F+F--F+F--F+F'
```

Exercice 4 :

Compléter la fonction `evolution(axiome,regle,etape)` qui :

- reçoit en entrée :
  - axiome : une chaîne de caractères représentant le motif de départ
  - regle : une chaîne de caractères indiquant la règle d'évolution
  - etape : un entier indiquant le numéro de l'étape à calculer
- renvoie une chaîne de caractères représentant la figure à l'étape demandée.

Par exemple : `evolution('F','F+' ,4)` renvoie `'F++++'`

In [17]:
```
def evolution(axiome, regle,etape):
    motif=axiome

    ...

    ...

    return motif
```

In [9]:
```
def evolution(axiome, regle,etape):
    motif=axiome
    for _ in range(etape):
        motif=suivant(motif,regle)

    return motif
```

In [10]:
```
assert evolution('F','F+',4)=='F++++'
```

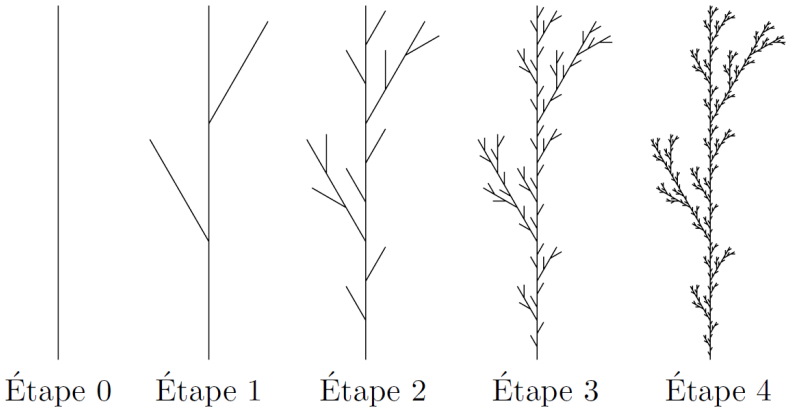
Exercice 5 :

En utilisant les fonctions précédentes, dessiner le flocon de Von Koch à l'étape 4.

In [15]:
```
#Flocon de Koch à l'étape 4
speed(0)
dessiner(3,60,evolution('F--F--F','F+F--F+F',4))
mainloop()
```

3. Gestions des ramifications

On souhaite à présent représenter une plante à l'aide d'un L-Système. Par exemple, avec une règle préalablement choisie et partant d'une branche F, on peut obtenir les images suivantes où le pas de rotation est de 20° (l'échelle est ajustée d'une image à l'autre et l'orienattion de départ initialisée à 90° pour un rendu plus réaliste) :



On se propose d'ajouter deux nouveaux symboles à l'alphabet des L-Systèmes :

- $[$: mémorise l'état de la tortue (coordonnées et orientation) au sommet d'une pile.
- $]$: dépile la dernière position de la tortue et replace la tortue à cet endroit (sans effectuer de tracé).

Par exemple, le motif de l'étape 1 est alors décrit par la chaîne $F[+F]F[-F]F$

Exercice 6 :

En prenant comme orientation de départ 90°, dessiner à la main :

- la figure $f(2, 45, F[-F][+F])$
- la figure $f(2, 90, F[-F[+F] - F]F)$:

Réponses :

- Y
- H

Exercice 7 :

Compléter la fonction `dessiner(unite,angle,motif)` pour qu'elle tienne compte de ces deux nouveaux symboles. Elle prend en paramètres :

- `unite` : la longueur
- `angle` : le pas de rotation a en degrés
- `motif` : une chaîne de caractères représentant le motif de la figure

Indications :

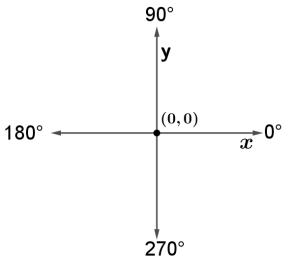
- La pile qui mémorise l'état de la tortue sera gérée par une liste python et par les méthodes `append` et `pop` . Chaque élément de cette pile est un tuple de deux valeurs:
  - La première valeur est elle-même un tuple contenant un couple de coordonnées.
  - La deuxième valeur est un entier qui représente un cap, en degrés.
  - Exemple :

[((15,58),90)), `((-20,115),70))`]

`↑ sommet de la pile`

Quelques commandes `turtle` :

- `heading()` : renvoie le cap en degrés de la tortue
- `setheading(cap)` : oriente la tortue au cap passé en argument, en degrés
- `position()` : renvoie un tuple qui contient les coordonnées cartésiennes x et y de la tortue.
- `goto(x,y)` : déplace la tortue aux coordonnées cartésiennes x et y passées en arguments
- `up()` : relève le stylo
- `down()` : abaisse le stylo



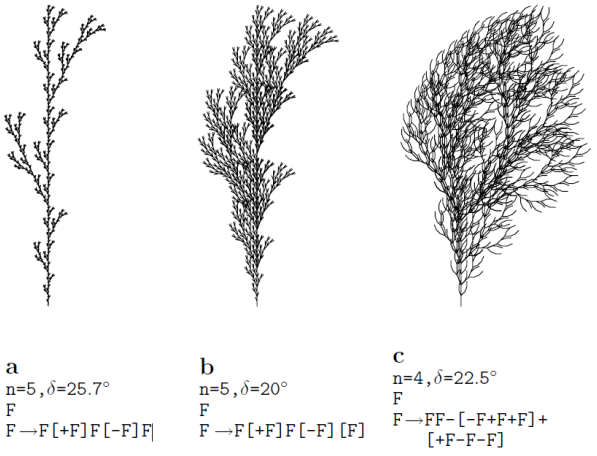
In [16]:
```
def dessiner(unite, angle, motif) :
    pile=[]
    for car in motif:
        if car=='F':
            down()
            forward(unite)
        elif car=='+':
            left(angle)
        elif car=='-':
            right(angle)
        elif car=='[':
            ...
        elif car==']':
            pos,cap=pile.pop()
            ...
            ...
            ...
```

In [11]:
```
def dessiner(unite, angle, motif) :
    pile=[]
    for car in motif:
        if car=='F':
            down()
            forward(unite)
        elif car=='+':
            left(angle)
        elif car=='-':
            right(angle)
        elif car=='[':
            pile.append((position(),heading()))
        elif car==']':
            pos,cap=pile.pop()
            up()
            goto(pos)
            setheading(cap)
```

In [12]:
```
#tests
hideturtle()
setheading(90)
dessiner(20,90,'F[-F[+F]-F]F')
mainloop()
```

Exercice 8 :

- L'image ci-contre est extraite de l'ouvrage d'Aristid Lindenmayer "Algorithmic Beauty Of Plants".
- En utilisant les différentes fonctions écrites précédemment, dessiner les trois exemples présentés sur cette image.



In [13]:
```
AXIOME='F'
UNITE= 10

#fig a
REGLE='F[+F]F[-F]F'
N=5
A=25.7

#fig b
REGLE='F[+F]F[-F][F]'
N=5
A=20

#fig c
REGLE='FF-[-F+F+F]+[+F-F-F]'
N=4
A=22.5
```

In [18]:
```
#initialisaiton turtle
tracer(120)
up()
goto(0,-200)
left(90)
hideturtle()


#Exemples
MOTIF=evolution(AXIOME,REGLE,N)
dessiner(UNITE,A,MOTIF)

mainloop()
```

4. En savoir plus

Il ne s'agissait ici que d'un bref aperçu des L-Systèmes. Cette modélisation est très puissante et permet, au delà de l'intérêt scientifique, de pousser le réalisme visuel obtenu à des niveaux très impressionnants :

- Vidéo générée avec le logiciel Houdini utilisant les L-Systèmes : https://vimeo.com/356351056 (https://vimeo.com/356351056)
- Page wikipédia sur les L-sytèmes : https://fr.wikipedia.org/wiki/L-Syst%C3%A8me (https://fr.wikipedia.org/wiki/L-Syst%C3%A8me)