

Algorithmique et programmation

Cours de maths - 4ème

Un programme, c'est une recette très précise que l'ordinateur suit à la lettre. Dans ce chapitre de 4ème, tu apprends à **écrire un algorithme**, à utiliser des **variables**, des **conditions** et des **boucles**, puis à faire dessiner des figures à un lutin avec des blocs, comme dans un logiciel de programmation visuelle. Tu apprends aussi à **suivre pas à pas** ce que fait un programme avec un tableau de suivi.

1. Qu'est-ce qu'un algorithme ?

Algorithme Un algorithme est une **liste ordonnée d'instructions** qui permet de résoudre un problème ou de réaliser une tâche. L'ordre compte : si on change l'ordre, le résultat peut changer.

Il y a des algorithmes partout : monter un meuble, suivre un itinéraire, préparer des crêpes. Un algorithme destiné à une machine se compose toujours de trois parties.

Partie	Rôle	Exemple : calculer le double d'un nombre
Entrée	les informations de départ	on choisit un nombre n
Traitement	la suite d'instructions	multiplier n par 2
Sortie	la réponse obtenue	afficher le résultat

Du langage courant au programme On peut écrire un même algorithme de plusieurs façons : en **langage courant** (des phrases), en **pseudo-code** (des phrases courtes et normalisées) ou en **blocs** que l'on empile, comme dans un logiciel à blocs. Dans ce cours, on écrit les programmes en blocs, représentés par du texte, avec le retrait des lignes pour montrer ce qui est « à l'intérieur » d'une boucle ou d'une condition.

Exemple : le programme de calcul Choisir un nombre, lui ajouter 3, multiplier le résultat par 5, afficher le résultat. En pseudo-code :

```
demande un nombre et le ranger dans x
x ← x + 3
x ← x × 5
afficher x
```

Pour $x = 4$, on obtient $(4 + 3) \times 5 = 35$. Ce programme calcule $5(x + 3)$.

2. Les variables et l'affectation

Variable Une variable est une « boîte » qui porte un **nom** et qui contient une **valeur** (un nombre, un mot, vrai ou faux). Le programme retrouve la valeur grâce au nom de la boîte.

Affectation Affecter, c'est ranger une valeur dans une variable. On écrit $A \leftarrow 7$ (« A reçoit 7 »). Dans un logiciel à blocs, on utilise le bloc « mettre A à 7 » ou « ajouter 2 à A ».

Le signe $\leftarrow$ n'est pas le signe = L'écriture $A \leftarrow A + 2$ ne veut pas dire que A est égal à $A + 2$ (ce serait impossible en mathématiques !). Elle veut dire : « je calcule la valeur actuelle de A plus 2, puis je la range dans A ». L'ancienne valeur est **effacée**.

Le tableau de suivi des variables Pour savoir ce que fait un programme, on construit un tableau : une colonne par variable, une ligne par instruction. On note la valeur de chaque variable après chaque instruction.

Exemple Suivons ce programme :

```
A ← 6
B ← A + 4
A ← A × 2
B ← B - A
```

Instruction	A	B
$A \leftarrow 6$	6	?
$B \leftarrow A + 4$	6	10
$A \leftarrow A \times 2$	12	10
$B \leftarrow B - A$	12	-2

À la fin, A vaut 12 et B vaut -2. Remarque : B a été calculé avec l'ancienne valeur de A (6), avant que A change.

Initialiser Avant d'utiliser une variable, il faut lui donner une valeur de départ : c'est l'**initialisation**. Un compteur démarre souvent à 0, un total aussi.

3. Les instructions conditionnelles

Condition Une condition est une phrase qui est soit **vraie**, soit **fausse** (par exemple : $n > 10$, ou $\text{âge} = 12$). Avec le bloc « si ... alors ... sinon », le programme choisit entre deux chemins.

```

si note  $\geq$  10 alors
    afficher « Admis »
sinon
    afficher « À revoir »
fin du si

```

Si la note vaut 14, la condition est vraie : le programme affiche « Admis ». Si elle vaut 7, il affiche « À revoir ». Un seul des deux blocs est exécuté.

Les comparaisons possibles < (strictement inférieur), > (strictement supérieur), $\leq$, $\geq$, = (est égal à), $\neq$ (est différent de). On peut aussi emboîter des conditions : « sinon, si ... alors ... ».

Teste les valeurs frontières La différence entre > et $\geq$ ne se voit que pour la valeur exacte. Pour la condition « note > 10 », la note 10 donne « faux » ; pour « note $\geq$ 10 », elle donne « vrai ». Quand tu testes un programme, essaie toujours les valeurs pile à la frontière.

4. La boucle « répéter » (« pour »)

Boucle « pour » Quand on sait **combien de fois** une action doit être répétée, on utilise une boucle « répéter ... fois » ou « pour i allant de 1 à n ». Les instructions à répéter sont écrites avec un retrait.

Exemple : une somme

```

S ← 0
pour i allant de 1 à 4
    S ← S + i
fin du pour
afficher S

```

Tour	i	S
départ	-	0
1	1	1
2	2	3
3	3	6
4	4	10

Le programme affiche 10 : c'est la somme $1 + 2 + 3 + 4$.

Pourquoi des boucles ? Écrire quatre fois les mêmes lignes est long et on risque de se tromper. Avec « répéter 4 fois », le programme est plus court et plus lisible. C'est la règle d'or : **on n'écrit jamais plusieurs fois la même série d'instructions** quand une boucle suffit.

5. La boucle « tant que »

Boucle « tant que » On l'utilise quand on **ne sait pas à l'avance** combien de répétitions seront nécessaires. Le programme teste la condition **avant chaque tour** : tant qu'elle est vraie, il recommence ; dès qu'elle est fausse, il passe à la suite.

Exemple : quand la somme dépasse-t-elle 50 ?

```
S ← 0
n ← 0
tant que S ≤ 50
    n ← n + 1
    S ← S + 8
fin du tant que
afficher n
```

S prend les valeurs 8, 16, 24, 32, 40, 48, 56. Le programme s'arrête quand $S = 56$ (car $56 > 50$). Il a fait 7 tours et affiche **7**.

Attention à la boucle infinie Si rien dans la boucle ne modifie ce que la condition teste, la condition reste vraie pour toujours et le programme ne s'arrête jamais. Vérifie toujours que **la variable testée change à chaque tour**, dans le bon sens.

Choisir sa boucle Nombre de répétitions connu : boucle « répéter » ou « pour ». Nombre de répétitions inconnu, qui dépend d'une condition : boucle « tant que ».

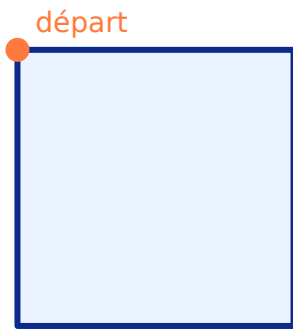
6. Dessiner avec un lutin

Un **lutin** est un petit personnage qui se déplace sur l'écran en laissant une trace quand le « stylo » est baissé. Ses deux déplacements de base sont **avancer de ... pas** et **tourner de ... degrés** (à droite ou à gauche). Il se déplace toujours dans la direction où il regarde.

Tracer un carré de 50 pas Un carré a 4 côtés égaux et 4 angles droits. À chaque sommet, le lutin tourne de 90° .

quand le drapeau vert est cliqué
stylo en position d'écriture

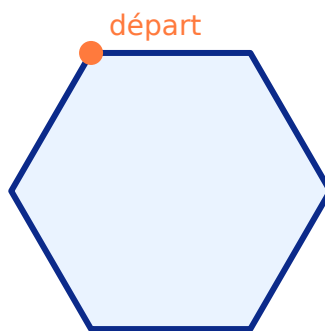
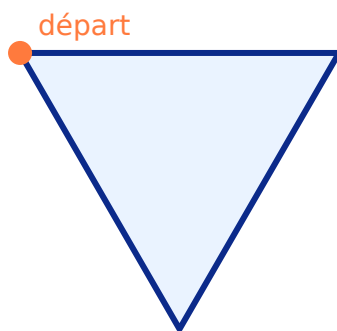
répéter 4 fois
avancer de 50 pas
tourner de 90 degrés vers la droite
fin du répéter



Le lutin revient à son point de départ (le point orange) et la figure est fermée.

Polygone régulier à n côtés Pour tracer un polygone régulier, on répète n fois : « avancer d'une longueur donnée » puis « tourner de $360^\circ \div n$ ». Le lutin fait un tour complet (360°) en tout.

Exemples Triangle équilatéral : $360 \div 3 = 120$, donc on tourne de 120° (et non de 60° , qui est l'angle *intérieur* du triangle). Hexagone régulier : $360 \div 6 = 60$, on tourne de 60° . Pentagone régulier : $360 \div 5 = 72$, on tourne de 72° .



Angle de rotation et angle de la figure Le lutin tourne de l'angle **extérieur**, pas de l'angle intérieur de la figure. Pour un carré, les deux valent 90° ; pour un triangle équilatéral, l'intérieur vaut 60° mais le lutin tourne de 120° (car $180^\circ - 60^\circ = 120^\circ$).

7. Blocs personnalisés et événements

Procédure (bloc personnalisé) Une procédure est un petit programme auquel on donne un **nom**. On peut ensuite l'appeler autant de fois qu'on veut, sans réécrire ses instructions. Une procédure peut recevoir des valeurs, appelées **paramètres** (par exemple la longueur d'un côté).

Exemple : deux carrés côte à côte

```
définir carré (longueur)
  répéter 4 fois
    avancer de longueur pas
    tourner de 90 degrés vers la droite
```

```
quand le drapeau vert est cliqué
carré (40)
avancer de 60 pas
carré (25)
```

Le même bloc « carré » sert deux fois, avec deux longueurs différentes.

Événement Un événement est quelque chose qui se produit pendant l'exécution : un clic sur le drapeau vert, une touche du clavier appuyée, un clic sur le lutin. Un script commence par un bloc « quand ... » : il se déclenche seulement quand l'événement a lieu. Plusieurs scripts peuvent s'exécuter en même temps.

```
quand la touche flèche droite est pressée
  ajouter 10 à x

quand la touche flèche haut est pressée
  ajouter 10 à y
```

Mathbot te conseille Un programme qui ne marche pas du premier coup, c'est normal ! Relis-le en jouant le rôle de l'ordinateur, avec un tableau de suivi : tu trouveras presque toujours l'erreur.

8. Tester et corriger un programme

Méthode pour corriger un programme 1) Prévois à la main le résultat attendu pour une petite valeur. 2) Exécute le programme avec cette valeur. 3) Si les deux résultats diffèrent, fais un tableau de suivi pour repérer la ligne fautive. 4) Corrige, puis teste de nouveau, y compris avec des valeurs frontières.

Voici les erreurs les plus fréquentes :

- **Oubli d'initialisation** : une variable sert avant d'avoir reçu une valeur.
- **Mauvais angle** : le lutin tourne de l'angle intérieur au lieu de l'angle extérieur.
- **Boucle infinie** : la condition du « tant que » ne change jamais.
- **Décalage d'un tour** : un « > » à la place d'un « $\geq$ », ou l'inverse.

- **Mauvais retrait** : une instruction placée dans la boucle alors qu'elle devrait être après (ou le contraire).

Exemple de correction On veut un hexagone mais le lutin trace une figure bizarre :

```
répéter 6 fois
  avancer de 40 pas
  tourner de 120 degrés vers la droite
```

L'angle est faux : pour 6 côtés, il faut tourner de $360 \div 6 = 60$ degrés. On remplace 120 par 60.

À retenir

- Un algorithme est une liste ordonnée d'instructions : **entrées, traitement, sorties**.
- Une variable est une boîte nommée qui contient une valeur ; **$A \leftarrow A + 2$** range dans A l'ancienne valeur plus 2.
- Le **tableau de suivi** donne la valeur des variables après chaque instruction.
- « si ... alors ... sinon » choisit entre deux chemins selon une condition vraie ou fausse ; teste les valeurs frontières.
- Boucle « répéter » ou « pour » : nombre de tours connu. Boucle « tant que » : on répète tant que la condition est vraie.
- Dans une boucle « tant que », une variable testée doit changer à chaque tour, sinon la boucle est infinie.
- Polygone régulier à n côtés : répéter n fois « avancer » et « tourner de $360^\circ \div n$ ».
- Une procédure évite de répéter les mêmes instructions ; un événement déclenche un script.