

# Pensée informatique et algorithmes

## Cours de maths - 5ème

Un ordinateur ne devine rien : il fait exactement ce qu'on lui demande, dans l'ordre. Dans ce chapitre de 5ème, tu apprends à **écrire des algorithmes**, à lire et à comprendre des **programmes en blocs** (comme dans un logiciel de programmation visuelle), à utiliser des **variables**, des **boucles** et des **conditions**, et à faire dessiner des figures à un petit personnage appelé le **lutin**.

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

**Algorithme** Un algorithme est une **suite ordonnée d'instructions** qui permet de réaliser une tâche ou de résoudre un problème. Si on change l'ordre des instructions, le résultat peut changer.

On en rencontre partout : une recette de gâteau, le mode d'emploi d'un jeu, un itinéraire pour aller au collège. Un algorithme destiné à une machine comporte presque toujours trois temps.

| Temps      | Rôle                                 | Exemple : calculer le triple d'un nombre |
|------------|--------------------------------------|------------------------------------------|
| Entrée     | on fournit une information de départ | on choisit un nombre                     |
| Traitement | on enchaîne des opérations           | on multiplie ce nombre par 3             |
| Sortie     | on obtient et on affiche la réponse  | on annonce le résultat                   |

**Trois façons d'écrire un même algorithme** En **langage courant** (des phrases), en **pseudo-code** (des phrases courtes et très précises) ou avec des **blocs** que l'on empile. Dans ce cours, les blocs sont représentés par du texte : une ligne par bloc, et un **retrait** (un décalage vers la droite) pour montrer ce qui est « à l'intérieur » d'une boucle ou d'une condition.

**Exemple : un programme de calcul** Choisir un nombre, lui ajouter 4, multiplier le résultat par 6, annoncer le résultat. En blocs :

```

quand le drapeau vert est cliqué
demander « Quel nombre ? » et attendre
mettre n à réponse
dire (n + 4) × 6
  
```

Pour un nombre choisi égal à 3, le lutin dit  $(3 + 4) \times 6 = 42$ .

## 2. Instructions, programme et lutin

**Instruction** Une instruction est une **action élémentaire** que le lutin (ou l'ordinateur) sait exécuter : avancer, tourner, dire un mot, poser une question, ajouter un nombre...

**Programme** Un programme est un **ensemble d'instructions organisées** qui réalise une tâche. Le lutin les exécute **une par une, dans l'ordre**, de haut en bas. Un programme commence souvent par un bloc d'événement, par exemple « quand le drapeau vert est cliqué ».

Les instructions se rangent en grandes familles, repérées par une couleur dans le logiciel : **mouvement** (avancer, tourner), **apparence** (dire, penser), **événements** (quand le drapeau est cliqué), **contrôle** (répéter, si... alors), **capteurs** (demander, toucher une touche), **opérateurs** (calculs), **variables**.

**Instruction d'entrée et instruction de sortie** Une instruction d'**entrée** récupère une information venant de l'utilisateur : poser une question, tester si une touche est appuyée. La réponse tapée est rangée dans une variable spéciale appelée **réponse**. Une instruction de **sortie** produit un effet visible ou audible : afficher une phrase, un résultat, déplacer le lutin, jouer un son.

### Exemple : un petit programme

```
quand le drapeau vert est cliqué  
demander « Comment t'appelles-tu ? » et attendre  
dire (Bonjour + réponse) pendant 2 secondes
```

La première instruction après l'événement est une instruction d'**entrée** ; la dernière est une instruction de **sortie**. Si tu écris Inès, le lutin affiche « Bonjour Inès ».

**L'ordre est important** Si on place « dire ... » avant « demander ... », le lutin parlera avant d'avoir reçu la réponse : il utilisera une réponse vide ou périmée. Quand un programme ne fait pas ce que tu attendais, vérifie d'abord l'ordre de ses instructions.

## 3. Programmes de calcul et expressions

Un programme de calcul se traduit par une **expression mathématique**. Il faut respecter les **priorités opératoires** : parenthèses d'abord, puis multiplications et divisions, puis additions et soustractions.

### Traduire un programme de calcul

Fiche en ligne : [maths-facile.eu/pensee-informatique-et-algorithmes-cours-5eme/](https://maths-facile.eu/pensee-informatique-et-algorithmes-cours-5eme/) · Cours, exercices et contrôles

1. Note le nombre de départ avec une lettre (par exemple  $n$ ).
2. Écris le résultat de chaque instruction l'une après l'autre.
3. Mets des **parenthèses** quand une opération doit être faite avant une autre qui a la priorité.

**Exemple** « Choisir un nombre  $n$  ; lui ajouter 5 ; multiplier le résultat par 2 ; retrancher 3. » Expression :  $(n + 5) \times 2 - 3$ . Pour  $n = 4$  :  $(4 + 5) \times 2 - 3 = 9 \times 2 - 3 = 18 - 3 = 15$ .

**Attention aux parenthèses**  $(n + 5) \times 2$  n'est pas égal à  $n + 5 \times 2$ . Pour  $n = 4$ , la première vaut 18 et la seconde vaut  $4 + 10 = 14$ . Dans un bloc d'opérations, les parenthèses se lisent comme « à l'intérieur du bloc, on calcule en premier ».

## 4. Les variables

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

**Affectation** Affecter, c'est ranger une valeur dans une variable. Dans un logiciel à blocs : « mettre *score* à 0 » (on la fixe) ou « ajouter 1 à *score* » (on la modifie à partir de sa valeur actuelle). L'ancienne valeur est **effacée**.

**Le tableau de suivi** Pour comprendre 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

```
mettre A à 5
mettre B à A + 2
ajouter 3 à A
mettre B à B × A
```

| Instruction      | A | B  |
|------------------|---|----|
| mettre A à 5     | 5 | ?  |
| mettre B à A + 2 | 5 | 7  |
| ajouter 3 à A    | 8 | 7  |
| mettre B à B × A | 8 | 56 |

À la fin, A vaut 8 et B vaut 56. Remarque : B a été calculé avec l'ancienne valeur de A (5), avant que A augmente.

**Initialiser** Avant de modifier une variable (par exemple de compter des points), il faut lui donner une valeur de départ : « mettre *score* à 0 ». Sinon on ne sait pas d'où l'on part.

## 5. Le lutin dans un repère

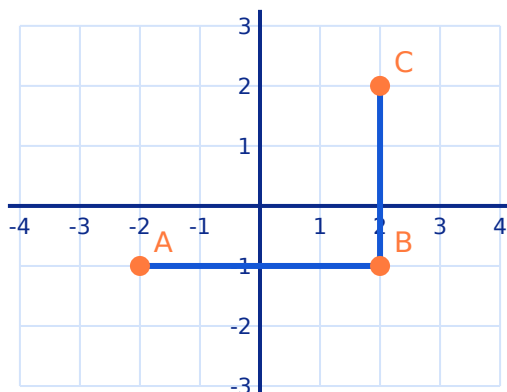
Le lutin se déplace dans un **repère** : un point repéré par deux nombres, l'**abscisse**  $x$  (position horizontale) et l'**ordonnée**  $y$  (position verticale). Le centre de l'écran est le point  $(0 ; 0)$ .

### Les instructions de mouvement

- **avancer de 30 pas** : le lutin se déplace de 30 pas dans la direction où il regarde.
- **tourner à droite (ou à gauche) de 90 degrés** : il pivote sur place, sans se déplacer.
- **aller à  $x : 2$   $y : -1$**  : il saute directement au point  $(2 ; -1)$ .
- **stylo en position d'écriture** : il laisse une trace ; **relever le stylo** : il se déplace sans tracer.

Au départ, on suppose que le lutin regarde vers la droite. Voici le chemin laissé par ce programme :

```
quand le drapeau vert est cliqué  
aller à  $x : -2$   $y : -1$   
stylo en position d'écriture  
aller à  $x : 2$   $y : -1$   
aller à  $x : 2$   $y : 2$ 
```



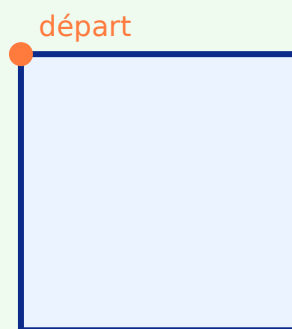
**Avancer et tourner, ou aller à ?** « Aller à » donne la position exacte de chaque sommet, mais il faut connaître les coordonnées. « Avancer » et « tourner » ne demandent que des longueurs et des angles : c'est la méthode la plus utile pour tracer des figures régulières.

## 6. Les boucles

**Boucle** Une boucle est une instruction qui **répète plusieurs fois** une ou plusieurs instructions, placées à l'intérieur (avec un retrait). Dans ce chapitre, on utilise « **répéter ... fois** » : on connaît à l'avance le nombre de répétitions.

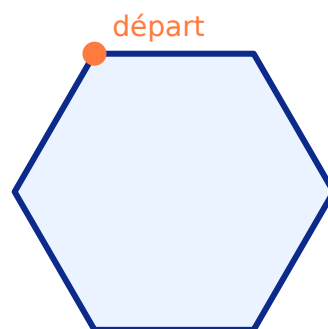
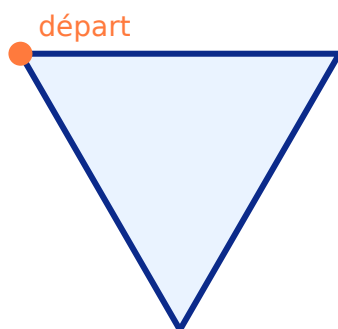
**Exemple : le carré** Sans boucle, il faut écrire 8 instructions. Avec une boucle, on n'en écrit que 3 :

```
répéter 4 fois  
  avancer de 50 pas  
  tourner à droite de 90 degrés
```



**Figures régulières** Pour tracer un polygone régulier à  $n$  côtés, on répète  $n$  fois : « avancer de la longueur du côté, puis tourner de  $360 \div n$  degrés ». Le lutin fait ainsi un tour complet ( $360^\circ$ ) en revenant à son point de départ.

| Figure               | Répéter | Angle de rotation | Calcul       |
|----------------------|---------|-------------------|--------------|
| triangle équilatéral | 3 fois  | $120^\circ$       | $360 \div 3$ |
| carré                | 4 fois  | $90^\circ$        | $360 \div 4$ |
| hexagone régulier    | 6 fois  | $60^\circ$        | $360 \div 6$ |



**L'angle de rotation n'est pas l'angle de la figure** Pour le triangle équilatéral, les angles de la figure mesurent  $60^\circ$ , mais le lutin tourne de  **$120^\circ$**  à chaque sommet

(l'angle « extérieur »). Le lutin tourne de  $360 \div 3 = 120^\circ$ , pas de  $60^\circ$ .

**Périmètre** Le nombre total de pas avancés par la boucle donne le périmètre de la figure : pour l'hexagone de côté 40,  $6 \times 40 = 240$  pas.

## 7. Les conditions

**Condition** Une instruction conditionnelle permet au programme de **choisir**. « **si test alors ... sinon ...** » : si le test est vrai, on exécute les instructions du « alors » ; s'il est faux, on exécute celles du « sinon ». Un test se fait avec une comparaison :  $<$  (plus petit que),  $>$  (plus grand que),  $=$  (égal à).

### Exemple

```
demander « Quelle est la température ? » et attendre
si réponse < 5 alors
    dire « Mets ton manteau ! »
sinon
    dire « Tu peux sortir en pull. »
```

Pour  $2^\circ\text{C}$ , le test est vrai : le lutin dit « Mets ton manteau ! ». Pour  $5^\circ\text{C}$ , le test «  $5 < 5$  » est faux : le lutin dit « Tu peux sortir en pull. ». Pour  $18^\circ\text{C}$ , même réponse.

**Les cas limites** La valeur exacte du seuil (ici 5) est souvent l'occasion d'une erreur. « Plus petit que 5 » ne contient pas 5. Teste toujours la valeur du seuil.

**Tester un programme à la main** Choisis plusieurs valeurs : une **au-dessous** du seuil, **le seuil lui-même**, une **au-dessus**. Pour chacune, dis quel test est vrai, puis quelle branche s'exécute.

## 8. Analyser et corriger un programme

### Quand un programme ne marche pas

1. Relis les instructions **dans l'ordre**, comme le lutin, sans rien deviner.
2. Fais un **tableau de suivi** sur une petite valeur.
3. Compare ce que tu obtiens à ce que tu voulais.
4. Repère l'instruction fautive, corrige-la, puis **teste à nouveau**.

Les erreurs classiques : un mauvais angle ( $45^\circ$  au lieu de  $90^\circ$ ), un nombre de répétitions trop petit, une variable non initialisée, une instruction placée avant l'instruction dont elle dépend, ou un seuil mal choisi dans une condition.

**Exemple** Ce programme devait tracer un carré, mais la figure reste ouverte :

```
répéter 3 fois  
  avancer de 40 pas  
  tourner à droite de 90 degrés
```

Il ne trace que trois côtés (3 répétitions). Pour fermer le carré, il faut **répéter 4 fois**.

**Le conseil de Mathbot** « Un bon programmeur se trompe souvent... mais il teste tout de suite ! Essaie d'abord avec un tout petit nombre, ou avec un petit dessin, avant de lancer le grand programme. »

## À retenir

- Un **algorithme** est une suite ordonnée d'instructions ; l'**ordre** compte.
- Une instruction d'**entrée** demande une information à l'utilisateur ; une instruction de **sortie** affiche ou déplace quelque chose.
- Un **programme** s'exécute de haut en bas, une instruction après l'autre, souvent après un bloc d'événement.
- Un programme de calcul se traduit par une **expression** avec les bonnes parenthèses.
- Une **variable** a un nom et une valeur ; on la modifie par affectation, et on suit ses valeurs dans un **tableau de suivi**.
- Le lutin se déplace dans un **repère** : avancer, tourner, aller à, stylo.
- « **répéter ... fois** » évite de recopier les mêmes instructions ; polygone régulier à  $n$  côtés : répéter  $n$  fois, tourner de  $360 \div n$  degrés.
- « **si ... alors ... sinon ...** » permet de choisir ; on teste toujours le seuil.
- Pour corriger un programme : le lire dans l'ordre, le suivre à la main, tester.