

Initiation à l'algorithmique

Cours de maths - 6ème

Un jeu vidéo, un robot aspirateur, une application sur téléphone : tous suivent des instructions écrites par des humains. Dans ce chapitre, tu apprends à **écrire, lire, compléter et corriger de petits programmes** : déplacer un lutin, répéter une action, calculer avec une variable et faire des choix. Tout se fait avec du papier et un crayon : tu joues le rôle de l'ordinateur !

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

Algorithme et instruction Un **algorithme** est une suite **ordonnée** d'**instructions** qui permet de résoudre un problème ou de réaliser une tâche. Une **instruction** est un ordre simple et précis, qu'on peut exécuter sans réfléchir. Quand un algorithme est écrit dans un langage compris par un ordinateur, on parle de **programme**.

Une recette de cuisine Pour faire des crêpes : 1) mettre la farine dans un saladier ; 2) ajouter les œufs ; 3) verser le lait en remuant ; 4) laisser reposer la pâte ; 5) cuire chaque crêpe à la poêle. Chaque ligne est une instruction, et l'ordre est important : on ne cuit pas avant d'avoir fait la pâte !

Les programmes par blocs Dans les logiciels de programmation par blocs, chaque instruction est un bloc coloré. On empile les blocs les uns sous les autres, et l'ordinateur les exécute de haut en bas. Dans ce cours, on écrit les blocs en français, l'un sous l'autre.

2. Se repérer et se déplacer

Un **lutin** (un petit personnage) se déplace sur un quadrillage. Il s'arrête toujours sur un nœud, c'est-à-dire à un croisement de lignes. On repère un nœud avec deux nombres entre parenthèses : le premier se lit sur l'axe horizontal, le second sur l'axe vertical. Par exemple, (3 ; 2) est le nœud situé 3 graduations vers la droite et 2 vers le haut.

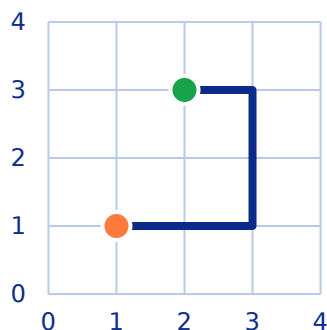
Déplacement absolu et déplacement relatif Un déplacement est **absolu** quand son effet ne dépend pas de la direction dans laquelle regarde le lutin : *Haut, Bas, Gauche, Droite* (ou nord, sud, ouest, est). Un déplacement est **relatif** quand son effet dépend de la direction du lutin : *Avancer, Tourner à droite de 90°, Tourner à gauche de 90°, Faire demi-tour*.

Même ordre, deux effets « Avancer de 2 » : si le lutin regarde vers la droite, il va 2

nœuds vers la droite ; s'il regarde vers le haut, il va 2 nœuds vers le haut. En revanche, « Haut » le fait toujours monter d'un nœud, quelle que soit sa direction.

Suivre un déplacement pas à pas 1) Repère le point de départ (et la direction du lutin si les ordres sont relatifs). 2) Exécute **une instruction à la fois**, en notant les coordonnées du nœud atteint. 3) Trace le trajet au crayon. 4) Lis la position finale.

Départ en (1 ; 1), instructions : Droite – Droite – Haut – Haut – Gauche. Le lutin passe par (2 ; 1), (3 ; 1), (3 ; 2), (3 ; 3), puis (2 ; 3).



Le point orange est le départ, le point vert est l'arrivée.

3. Une séquence : l'ordre compte

Séquence d'instructions Une **séquence** est une liste d'instructions exécutées l'une après l'autre, dans l'ordre où elles sont écrites.

Attention à l'ordre Avec des déplacements relatifs, changer l'ordre change tout. Pour un lutin qui regarde vers la droite : « Avancer – Tourner à gauche – Avancer » l'amène en haut à droite, alors que « Tourner à gauche – Avancer – Avancer » le fait monter de deux nœuds sans jamais aller à droite.

Produire une séquence pour un chemin donné Décompose le chemin en **segments** droits. Pour chaque segment, écris autant de « Haut », « Bas », « Gauche » ou « Droite » que de nœuds à parcourir. Relis ta séquence en suivant le chemin avec ton crayon pour la tester.

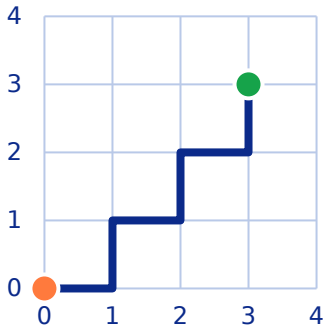
4. Les boucles : répéter pour écrire moins

Quand une même suite d'instructions revient plusieurs fois, on utilise une **boucle**.

Boucle « répéter » Le bloc « Répéter n fois » exécute n fois de suite les instructions placées à l'intérieur. Les instructions concernées sont **décalées vers la droite** pour montrer qu'elles sont dans la boucle.

Répéter 3 fois
Droite
Haut

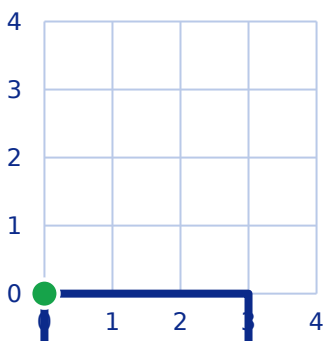
Ce programme équivaut à écrire : Droite – Haut – Droite – Haut – Droite – Haut. Il dessine un escalier, de (0 ; 0) à (3 ; 3).



Le point orange est le départ, le point vert est l'arrivée.

Dessiner un carré Un lutin qui regarde vers la droite doit tracer un carré de côté 3. Sans boucle, il faudrait 8 instructions ; avec une boucle, il n'en faut que 3 :

Répéter 4 fois
Avancer 3
Tourner à droite de 90°



Le point orange est le départ, le point vert est l'arrivée.

Boucles imbriquées On peut placer une boucle **dans** une autre boucle. On commence par lire la boucle intérieure : « Répéter 4 fois (Répéter 2 fois (Droite) puis Haut) » fait $4 \times (2 + 1) = 12$ déplacements.

Compter les instructions Le nombre d'instructions réellement exécutées est : (nombre de répétitions) $\times$ (nombre d'instructions dans la boucle). Pour « Répéter 5 fois (Avancer, Tourner à gauche de 90°) », cela fait $5 \times 2 = 10$ instructions exécutées.

5. Les variables et les programmes de calcul

Variable Une **variable** est une « boîte » qui porte un nom et qui contient une valeur (un nombre, par exemple). On peut lire cette valeur, ou la **modifier** avec l'instruction « mettre ... à ... » ou « ajouter ... à ... ».

```
mettre score à 0
Répéter 4 fois
    ajouter 5 à score
```

On suit les valeurs de la variable dans un tableau :

Après...	début	1 tour	2 tours	3 tours	4 tours
score	0	5	10	15	20

À la fin, la variable « score » contient 20.

Exécuter un programme de calcul Un programme de calcul est un algorithme dont les instructions sont des opérations. Pour l'exécuter, on **écrit le résultat après chaque ligne**.

Programme : choisir un nombre ; le multiplier par 4 ; ajouter 3. Avec 6 : $6 \times 4 = 24$, puis $24 + 3 = 27$.

Remonter le programme Pour savoir quel nombre il faut choisir pour obtenir un résultat donné, on fait le chemin inverse, en partant de la fin avec les opérations contraires. Pour obtenir 35 avec le programme ci-dessus : $35 - 3 = 32$, puis $32 \div 4 = 8$. Vérification : $8 \times 4 + 3 = 35$.

6. Les tests : si... alors... sinon

Instruction conditionnelle Un **test** permet à l'algorithme de faire un choix. Si la condition est vraie, on exécute les instructions du « alors » ; sinon, on exécute celles du « sinon » (ou rien s'il n'y a pas de « sinon »).

```
Si nombre est pair
alors
    diviser le nombre par 2
sinon
    ajouter 1 au nombre
```

Exécution Avec 18 : 18 est pair, donc on calcule $18 \div 2 = 9$. Avec 7 : 7 n'est pas pair,

donc on calcule $7 + 1 = 8$.

Strictement ou non ? « score > 10 » est vrai pour 11, mais faux pour 10. « score ≥ 10 » est vrai pour 10. Une seule petite différence de symbole peut changer le résultat du test : lis-le toujours avec soin.

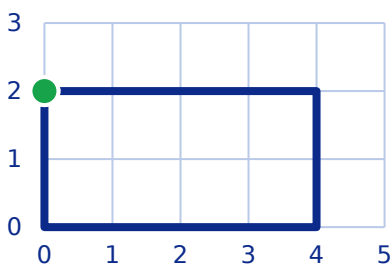
On peut enchaîner plusieurs tests : « Si âge < 12 alors tarif 4 €, sinon si âge < 60 alors tarif 8 €, sinon tarif 5 € ». On teste dans l'ordre et on s'arrête au premier test vrai.

7. Compléter un programme

Trouver les éléments manquants 1) Regarde le résultat attendu (la figure ou le nombre final). 2) Repère la partie qui se répète : c'est le contenu de la boucle. 3) Calcule le nombre de répétitions. 4) Teste ton programme complété en l'exécutant à la main.

Exemple On veut un lutin qui trace un rectangle de 4 sur 2, en regardant vers la droite au départ. Le motif « Avancer 4, tourner à droite, Avancer 2, tourner à droite » fait deux côtés ; il faut le répéter **2 fois**.

```
Répéter ... fois
  Avancer 4
  Tourner à droite de 90°
  Avancer 2
  Tourner à droite de 90°
```



Le point orange est le départ, le point vert est l'arrivée.

8. Déboguer un programme

Bogue Un **bogue** (ou bug) est une erreur dans un programme : le programme s'exécute, mais le résultat n'est pas celui qu'on attendait. **Déboguer**, c'est chercher et corriger cette erreur.

La chasse aux bogues 1) Écris ce que le programme devrait faire. 2) Exécute-le à la main, instruction par instruction, en notant le résultat de chaque étape. 3) Repère la

première ligne où ton résultat s'écarte du résultat attendu. 4) Corrige cette ligne (un nombre, un sens de rotation, un ordre, un nombre de répétitions) puis teste à nouveau.

Mathbot te conseille « Un programmeur qui se trompe n'est pas un mauvais programmeur : c'est un programmeur qui teste ! Quand ton programme ne fait pas ce que tu veux, change une seule chose à la fois, et teste après chaque changement. »

À retenir

- Un algorithme est une suite ordonnée d'instructions ; l'ordre des instructions est essentiel.
- Un déplacement absolu (Haut, Bas, Gauche, Droite) ne dépend pas de la direction du lutin ; un déplacement relatif (Avancer, Tourner) en dépend.
- La boucle « Répéter n fois » exécute n fois les instructions qu'elle contient ; on peut en imbriquer plusieurs.
- Une variable est une boîte qui contient une valeur ; on suit ses changements dans un tableau.
- Un test « si... alors... sinon » permet de choisir selon une condition ; attention à $<$ et $\leq$.
- Pour déboguer, on exécute à la main et on cherche la première ligne qui s'écarte du résultat attendu.