CINQUIÈME

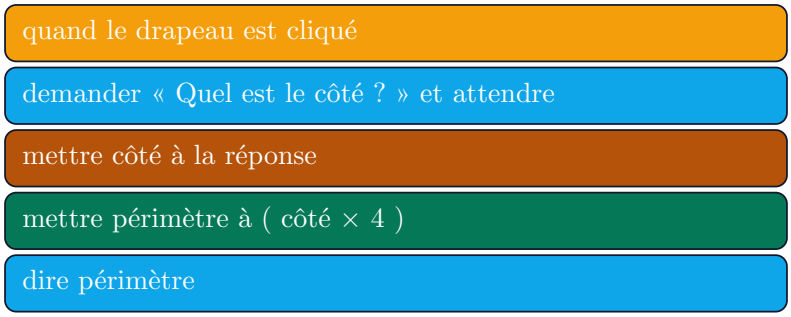
Programmation par blocs

I - Instructions, séquences, entrées et sorties

Définition 1
Une **instruction** est un ordre élémentaire, compris sans ambiguïté par la machine. Une **séquence** est une suite d'instructions exécutées **dans l'ordre**, de haut en bas.

Vocabulaire
Dans un langage de programmation par blocs, chaque instruction est un bloc. Les blocs s'emboitent verticalement : l'empilement **est** l'ordre d'exécution.

Définitions
- Les **entrées** d'un programme sont les informations qu'il reçoit de l'extérieur : une valeur saisie, un clic, un nombre au hasard.
- Les **sorties** sont ce qu'il produit : un affichage, un dessin, un son.



l'entrée est la réponse saisie ; la sortie est ce que le programme dit

Fig. 1. – Un programme qui demande un côté, calcule le périmètre d'un carré, puis l'affiche : l'entrée est la réponse saisie, la sortie est le nombre dit.

Définition 2
Une **variable** est une case mémoire portant un nom, dans laquelle le programme range une valeur pour la réutiliser plus tard.

Remarque
L'instruction « mettre x à 7 » n'est pas une égalité mathématique : c'est un ordre de rangement. La valeur précédente de x est perdue.

Méthode 1 — Analyser un programme donné

1. Repérer ses entrées : que doit-on lui fournir ?
2. Exécuter la séquence pas à pas, en notant après chaque bloc la valeur de chaque variable.
3. Lire la sortie obtenue et la confronter à ce qu'on attendait.

Exemple
Avec le programme ci-dessus et la réponse 7 :

Après l'instruction	côté	périmètre
mettre côté à la réponse	7	—
mettre périmètre à (côté × 4)	7	28
dire périmètre	7	28

La sortie est 28.

II - Formules et expressions informatiques

Définition 3
Une **expression informatique** est la traduction d'une formule mathématique dans le langage : les nombres, les variables et les opérateurs y sont emboités les uns dans les autres.

Exemples

Formule	Expression en blocs
$P = 4 \times c$	`( côté × 4 )`
$P = 2 \times (L + l)$	`( 2 × ( longueur + largeur ) )`
$\mathcal{A} = \frac{b \times h}{2}$	`( ( base × hauteur ) / 2 )`

Remarque
Les blocs d'opération ne connaissent pas les priorités : l'ordre des calculs est celui de l'emboitement. Ce sont les blocs eux-mêmes qui jouent le rôle des parenthèses.

Méthode 2 — Prévoir la valeur d'une expression
1. Remplacer chaque variable par sa valeur actuelle.
2. Évaluer les blocs les plus intérieurs d'abord, puis remonter.
3. Comparer ensuite avec ce que le programme affiche : un écart signale une erreur d'emboitement.

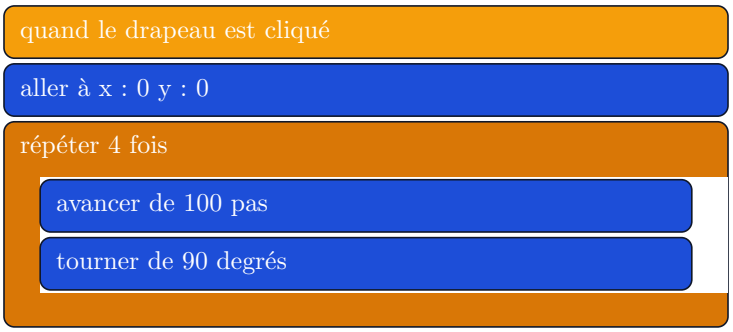
Exemple
Avec longueur = 8 et largeur = 3, l'expression `( 2 × ( longueur + largeur ) )` vaut :

$$2 \times (8 + 3) = 2 \times 11 = 22.$$

Emboitée autrement, `( ( 2 × longueur ) + largeur )` vaudrait 19 : ce ne serait plus le périmètre.

III - Boucle inconditionnelle

Définition 4
Une **boucle inconditionnelle** répète une séquence d'instructions un nombre de fois fixé à l'avance : « répéter n fois ».



la boucle exécute quatre fois de suite les deux mêmes instructions

Fig. 2. – Les deux instructions placées dans la boucle sont exécutées quatre fois de suite : le lutin trace un carré.

Méthode 3 — Repérer ce qui se répète
1. Écrire la séquence complète, sans boucle.
2. Repérer le motif qui revient à l'identique, et compter combien de fois.
3. Remplacer les répétitions par une boucle « répéter n fois » contenant ce seul motif.

Exemple
Pour tracer un carré, la séquence « avancer, tourner » revient quatre fois : huit blocs deviennent une boucle de deux blocs. Pour un triangle équilatéral, ce serait « répéter 3 fois : avancer de 100 ; tourner de 120 degrés », car $360 \div 3 = 120$.

Propriété 1
Pour tracer un polygone régulier à n côtés, on répète n fois « avancer, puis tourner de $360 \div n$ degrés » : le lutin fait un tour complet.

Remarque
Modifier le seul paramètre n change entièrement le dessin sans toucher au reste du programme. C'est tout l'intérêt d'une boucle : le motif est écrit une fois, exécuté autant de fois qu'on veut.