Introduction

Un ordinateur peut faire bien des choses,
mais il faut d 'abord les lui expliquer.

Apprendre la programmation, ce n'est pas seulement savoir écrire un programme, c'est aussi comprendre de quoi il est fait, comment il est fait et ce qu'il fait.

Un programme est essentiellement constitué d'expressions et d'instructions.

Nous introduisons dans ce chapitre les trois premières instructions fondamentales que sont l'affectation, la séquence et le test.

Pour mettre en évidence leur structure, nous indentons les programmes et utilisons les accolades lorsque l'écriture est ambiguë.

Nous étudions les instructions en observant les transformations qu'elles opèrent sur l'état de l'exécution du programme, c'est-à-dire sur l'ensemble des boîtes pouvant contenir des valeurs, avec leur nom et leur valeur, le cas échéant.

John Backus (1924-2007) est l'auteur de l'un des premiers langages de programmation : le langage Fortran (1954). Il a par la suite proposé, avec Peter Naur, la notation de Backus et Naur qui permet de décrire des grammaires, en particulier celles des langages de programmation.

Grace Hopper (1906-1992) est, elle aussi, l'auteur d'un des premiers langages de programmation : le langage Cobol (1959).
Avant cela, elle a été l'une des premières à programmer le Harvard Mark I de Howard Aiken, l'un des tous premiers calculateurs électroniques.

Un premier programme

Voici un premier petit programme écrit en Java :

```
void main() {
  int a = 4;
  int b = 7;
  println("À vous de jouer");
  int x = readInt();
  int y = readInt();
  if (x == a && y == b) {
    println("Coulé");
  } else {
    if (x == a || y == b) {
      println("En vue");
    } else {
      println("À l'eau");
    }
  }
}
```

Quand on exécute ce programme, il affiche « À vous de jouer » puis attend que l'on tape deux nombres au clavier. Si ces deux nombres sont 4 et 7, il affiche « Coulé » ; si le premier est 4 ou le second 7, mais pas les deux, il affiche « En vue », sinon il affiche « À l'eau ».

Ce programme permet de jouer à la bataille navale, dans une variante simplifiée dans laquelle il n'y a qu'un seul bateau, toujours placé au même endroit et qui n'occupe qu'une seule case de la grille.

On considère qu'un bateau est « en vue » si la case proposée est sur la même ligne ou la même colonne que le bateau.

Un premier programme

Exercice 1 :
Modifier ce programme afin qu'il affiche À toi de jouer et non À vous de jouer.

Exercice 2 :
Modifier ce programme afin que le bateau soit sur la case de coordonnées (6 ; 9).

Exercice 3 :
En général, à la bataille navale, un bateau n'est « en vue » que si la case touchée est immédiatement voisine de celle du bateau.
Modifier le premier programme pour tenir compte de cette règle.
On pourra traiter le cas où les cases diagonalement adjacentes au bateau sont « en vue » et le cas où elles ne le sont pas.

Un premier programme

Correction exercice 3 :

```
void main() {
  int a = 4;
  int b = 7;
  println("À vous de jouer");
  int x = readInt();
  int y = readInt();
  int diff_x,diff_y;
  diff_x = Math.abs(x - a);
  diff_y = Math.abs(y - b);
  if (diff_x == 0 && diff_y == 0) println("Coulé");
  else if (diff_x <= 1 && diff_y <= 1) println("En vue");
  else println("À l'eau");
}
```

Commentaires :

- **On calcule la différence en ligne Math.abs(x-a) et en colonne Math.abs(y-a) entre la ligne du bateau et la valeur saisie en x et la colonne du bateau et la valeur saisie en y.**

Les ingrédients d'un programme

Le programme de bataille navale utilise des instructions de différentes formes :

- des *affectations* de la forme v = e; où v est une variable et e une expression,
- des *instructions d'entrée* de la forme v = readInt(); où v est une variable,
- des *instructions de sortie* de la forme println(e); où e est une expression,
- des *séquences* de la forme p q (c'est-à-dire p suivi de q) où p et q sont deux instructions,
- des *tests* de la forme if (e) p else q où e est une expression et p et q deux instructions.

Les ingrédients d'un programme

Exercice 4 :
Que fait ce programme ?

```
a = readInt();
b = readInt();
c = readInt();
d = readInt();
if (b == 0 || d == 0)
{
        println("Dénominateur nul interdit !");
}
else
{
        println(a * d + c * b);
        println(b * d);
}
```

Les ingrédients d'un programme

Correction exercice 4 :

Ce programme calcule le numérateur et le dénominateur de la somme des fractions a/b + c/d.

Les ingrédients d'un programme

Exercice 5 :
L'exécution du programme :

```
x = 4;
y = x + 1;
x = 10;
println(y);
```

produit-elle l'affichage de la valeur 5 ou de la valeur 11 ?

Exercice 6 :
a) Écrire un programme qui, étant donné une équation du second degré, détermine le nombre de ses solutions réelles et leurs valeurs éventuelles.

Les ingrédients d'un programme

Correction exercice 5 :

Le programme affiche 5 car lors de l'affection y = x + 1
la valeur de x est 4.

Les ingrédients d'un programme

Correction exercice 6 :

```
void main() {
   double a = readDouble("Entrer le coefficient a : ");
   double b = readDouble("Entrer le coefficient b : ");
   double c = readDouble("Entrer le coefficient c : ");
   double delta;
   delta = b * b - 4 * a * c;
   if (delta < 0.0) {
      println("Pas de solution dans R");
   } else {
      if (delta == 0.0) {
         print("Une solution : ");
         println( - b / (2 * a));
      } else {
         print("Deux solutions : ");
         print(( - b - Math.sqrt(delta)) / (2 * a));
         print(" et ");
         println(( - b + Math.sqrt(delta)) / (2 * a));
      }
   }
}.
```

Les ingrédients d'un programme

Exercice 7 :

a) Essayer le programme précédent avec les entrées :

- a = 1.0, b = 0.0, c = 1.0E-10
- a = 1.0, b = 0.0, c = -1.0E-10.

Montrer qu'une infime variation sur l'un des coefficients permet de franchir la ligne qui sépare les cas où l'équation a des solutions des cas où elle n'en a pas.

b) Essayer le programme ci-dessus avec les entrées :

- a = 1.0, b = 6.0, c = 9.0
- a = 0.1, b = 0.6, c = 0.9.

Expliquer les résultats.

Les ingrédients d'un programme

Correction exercice 7 :

a)

- Pour a = 1.0, b = 0.0, c = 1.0E-10 : le programme affiche :
  « pas de solution dans R »
  delta = b^2 - 4ac = -4E-10 < 0 : effectivement l'équation n'a pas de solution.

- Pour a = 1.0, b = 0.0, c = -1.0E-10 : le programme affiche :
  « deux solutions : -1.0E-5 et 1.0E-5 » ce qui est correct.

b)

- Pour a = 1.0, b = 6.0, c = 9.0, le programme affiche :
  « Une solution : -3.0 » ce qui est correct.
- Pour a = 0.1, b = 0.6, c = 0.9, le programme affiche :
  « Pas de solution dans R » ce qui est faux car cette équation a la même solution unique que la précédente : -3.

L'explication réside dans le calcul du delta :
Le programme calcule delta = $0{,}6^2$ - 4×0,9 = -5.551115123125783E-17
Au lieu de 0.
C'est un problème connu quand on manipule des nombres à virgule (double ou float)

La mise au point d'un programme

Pour vérifier si un programme ne produit pas d'erreur au cours de son exécution et s'il effectue réellement la tâche que l'on attend de lui, une première méthode consiste à exécuter plusieurs fois ce programme, en lui fournissant des entrées, appelées *tests*, qui permettent de détecter les erreurs éventuelles.

Pour qu'elles jouent leur rôle, il faut choisir ces entrées de sorte que :

- on sache quelle doit être la sortie correcte du programme avec chacune de ces entrées,
- chaque cas distinct d'exécution du programme soit parcouru avec au moins un choix d'entrées,
- les cas limites soient essayés : par exemple le nombre 0, la chaîne vide ou à un seul caractère.

La mise au point d'un programme

Exercice 8

Proposer un jeu de tests satisfaisant pour le programme de bataille navale.

Exercice 9

Proposer un jeu de tests satisfaisant pour le programme de calcul des solutions réelles d'une équation du second degré.

La mise au point d'un programme

Correction exercice 8

Au minimum, il faut vérifier que le bateau est effectivement coulé si l'on donne les bonnes coordonnées, mais non coulé si l'on en donne de mauvaises. Par ailleurs, il faut tester si le programme affiche correctement En vue, *et donc tester au moins une case dans la même colonne que le bateau et une case dans la même ligne. Ces deux derniers tests permettront également de vérifier que les instructions conditionnelles du programme sont écrites correctement, et que par exemple il ne suffit pas d'avoir trouvé la bonne ligne pour couler le bateau. On testera donc le programme sur les entrées suivantes, par exemple, avec les résultats attendus :*

- *(4 ; 7) :* Coulé,
- *(1 ; 2) :* À l'eau,
- *(4 ; 9) :* En vue *(même ligne),*
- *(8 ; 7) :* En vue *(même colonne).*

On pourrait également tester ce qu'il se passe si l'on entre une coordonnée décimale ou une coordonnée qui dépasse les limites du tableau de jeu.

La mise au point d'un programme

Correction exercice 9

Il faut proposer au minimum un jeu de tests comportant les trois cas de signe pour le discriminant :

- $\Delta < 0$
- $\Delta = 0$
- $\Delta > 0$

Les instructions et les expressions

L'affectation x = y + 3; est une instruction.

Elle est composée d'une variable x et d'une *expression* y + 3.

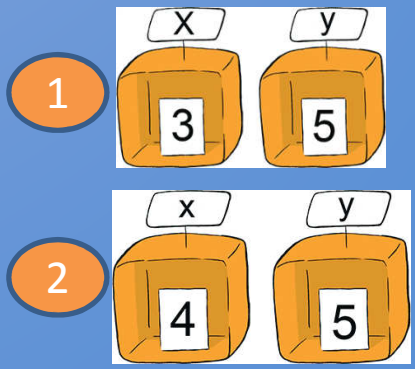
On attribue une *valeur* à chaque expression.

Pour les expressions sans variables, comme (2 + 5) * 3, dont la valeur est 21, la valeur s'obtient simplement en effectuant les opérations présentes dans l'expression, dans cet exemple une addition et une multiplication.

La valeur d'une expression qui contient des variables, par exemple (2 + x) * 3, se définit de la même manière, mais dépend de l'état dans lequel on calcule cette valeur.

Par exemple, la valeur de l'expression (2 + x) * 3

- dans l'état 1 est 15,
- dans l'état 2 est 18.



Les instructions et les expressions

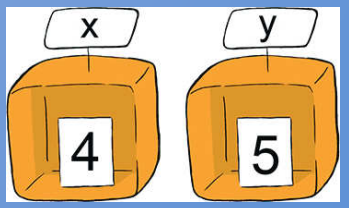
Exercice 10

Les suites de symboles suivantes sont-elles des instructions ou des expressions ?

- x
- x = y;
- x = y + 3;
- x + 3;
- println(x + 3);
- x readInt();
- x == a
- x == a && y == b

Exercice 11

Déterminer la valeur des expressions suivantes dans l'état 1 :



- y + 3
- x + 3
- x + y
- x * x
- y == 5
- x == 3 && y == 5

Les instructions et les expressions

Correction exercice 10

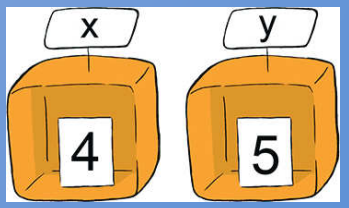
Les suites de symboles suivantes sont-elles des instructions ou des expressions ?

- x : expression
- x = y; : instruction
- x = y + 3; : instruction
- x + 3; : expression
- println(x + 3); : instruction
- x = readInt(); : instruction
- x == a : expression
- x == a && y == b : expression

Correction exercice 11

Déterminer la valeur des expressions suivantes dans l'état .

1



- y + 3 : 8
- x + 3 : 7
- x + y : 9
- x * x : 16
- y == 5 : Vrai
- x == 3 && y == 5 : Faux

Les opérations

+	Addition entière, décimale et concaténation de chaînes de caractères
-	Soustraction entière, décimale
*	Multiplication entière, décimale
/	Quotient de la division euclidienne. Attention cette division est inhabituelle pour les nombres négatifs : -5 / 2 vaut -2 et non -3, et -5 / -2 vaut 2.
%	Reste de la division euclidienne. Attention encore aux nombres négatifs : -5 % 2 vaut -1 et -5 % -2 aussi.
Math.pow	Puissance
Math.sqrt	Racine
Math.PI	π
Math.sin	Sinus
Math.cos	Cosinus
Math.exp	Exponentielle
Math.log	Logarithme népérien
Math.abs	Valeur absolue
Math.min	Minimum
Math.max	Maximum
Math.floor	Partie entière
Math.random	Nombre aléatoire décimal entre 0 et 1, selon la loi uniforme

==	Égal. S'applique aux valeurs numériques et booléennes, mais pas aux chaînes de caractères ni aux tableaux.
!=	Différent. S'applique aux valeurs numériques et booléennes, mais pas aux chaînes de caractères ni aux tableaux.
<=	Inférieur ou égal.
<	Inférieur strictement.
>=	Supérieur ou égal.
>	Supérieur strictement.
!	Non.
&&	Et (Variante : &.)
||	Ou (Variante :|.)

Les opérations

Exercice 12

En utilisant la fonction Math.random, écrire un programme qui affiche aléatoirement pile ou face de façon équiprobable.

Correction exercice 12

```
void main() {
   double alea;
   alea = Math.floor(Math.random() * 2);
   if (alea == 0) println("Pile");
   else println("Face");
}
```

Les accolades

L' instruction if (x == 0) y = 1; else y = 2; z = 4; peut être construite :

- ou bien comme la séquence des instructions if (x == 0) y = 1; else y = 2; et z = 4;
- ou alors comme le test formé de l'expression x == 0 et des instructions y = 1; et y = 2; z = 4;

On lève cette ambiguïté en utilisant des accolades, et on écrit :

- {if (x == 0) y = 1; else y = 2;} z = 4; la première instruction
- if (x == 0) y = 1; else {y = 2; z = 4;} la seconde.

Quand on n'utilise pas d'accolades, l'instruction :

if (x == 0) y = 1; else y = 2; z = 4;

signifie :

- {if (x == 0) y = 1; else y = 2;} z = 4;
- et non if (x == 0) y = 1; else {y = 2; z = 4;}.

Les accolades

Exercice 13

Quel est le résultat de l'exécution des instructions :

1

```
if (x ==  4) {
y  =  1;}
else {
y  =  2;
z = 3;}
```

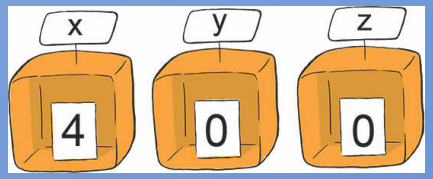
2

```
{
if (x ==  4) {
y  =  1;}
else {
y  =  2;}}
z = 3;
```

3

```
if (x ==  4) {
y  =  1;}
else {
y  =  2;}
z = 3;
```

Dans l'état :



Les accolades

Correction exercice 13

1

x = 4
y = 1
z = 0

2

x = 4
y = 1
z = 3

3

x = 4
y = 1
z = 3

Les accolades

Indenter un programme, c'est-à-dire insérer des espaces blancs en début de ligne, est un moyen de visualiser le niveau d'imbrication auquel une instruction se trouve.

On insère un espace blanc par accolade ouverte et non fermée située avant cette instruction dans le programme.

Dans certains langages, comme Python, l'indentation joue le rôle de parenthèses, comme les accolades en Java.

Dans certains langages, comme Python, l'indentation joue le rôle de parenthèses, comme les accolades en Java.

En Java, en revanche, l'indentation est une aide à la lecture, mais ne change pas la signification des programmes.