Introduction

À la source d'un programme

Aujourd'hui, l'ordinateur est un objet courant qui est aussi bien capable d'imprimer les factures des clients d'un magasin d'outillage que d'analyser des molécules chimiques très complexes.

Bien évidemment, ce n'est pas la machine-ordinateur en tant que telle qui possède toutes ces facultés, mais les applications ou encore les logiciels installés sur celle-ci. Pour émettre une facture, l'utilisateur doit faire appel à une application spécifique de facturation. Pour étudier des molécules, les chercheurs utilisent des logiciels scientifiques très élaborés.

Un ordinateur sans programme n'a aucune utilité. Seul le programme fait de l'ordinateur un objet « intelligent », traitant l'information de façon à en extraire des valeurs pertinentes selon son objectif final.

Ainsi, créer un programme, une application, c'est apporter de l'esprit à l'ordinateur. Pour que cet esprit donne sa pleine mesure, il est certes nécessaire de bien connaître le langage des ordinateurs, mais surtout, il est indispensable de savoir programmer. La programmation est l'art d'analyser un problème afin d'en extraire la marche à suivre, l'algorithme susceptible de résoudre ce problème.

C'est pourquoi ce chapitre commence par aborder la notion d'algorithme. À partir d'un exemple, tiré de la vie courante, nous déterminerons les étapes essentielles à l'élaboration d'un programme (« Construire un algorithme »). À la section suivante, « Qu'est-ce qu'un ordinateur ? », nous examinerons le rôle et le fonctionnement de l'ordinateur dans le passage de l'algorithme au programme. Nous étudierons ensuite, à travers un exemple simple, comment écrire un programme en ActionScript 3 et comment l'exécuter (« Un premier programme en ActionScript 3 »).

Enfin, dans la section « L'environnement de programmation Flash », nous donnons un aperçu détaillé de l'interface de Flash CS3 et Flash CS4/CS5, base nécessaire pour écrire des programmes en ActionScript 3.

Construire un algorithme

Un programme est écrit avec l'intention de résoudre une problématique telle que stocker des informations relatives à un client, imprimer une facture ou encore calculer des statistiques afin de prévoir le temps qu'il fera dans 2, 3 ou 5 jours.

Un ordinateur sait calculer, compter, trier ou rechercher l'information dans la mesure où un programmeur lui a donné les ordres à exécuter et la marche à suivre pour arriver au résultat.

Cette démarche s'appelle un « algorithme ».

Déterminer un algorithme, c'est trouver un cheminement de tâches à fournir à l'ordinateur pour qu'il les exécute. Voyons comment s'y prendre pour le construire.

Ne faire qu'une seule chose à la fois

Avant de réaliser une application concrète, telle que celle proposée en projet dans cet ouvrage (voir chapitre 1 « Traiter les données », section « Le projet mini site »), nécessairement complexe par la diversité des actions qu'elle doit réaliser, simplifions-nous la tâche en ne cherchant à résoudre qu'un problème à la fois.

Considérons que créer une application, c'est décomposer cette dernière en plusieurs sous-applications qui, à leur tour, se décomposent en micro-applications, jusqu'à descendre au niveau le plus élémentaire. Cette démarche est appelée « analyse descendante ». Elle est le principe de base de toute construction algorithmique.

Pour bien comprendre cette démarche, penchons-nous sur un problème réel et simple à résoudre : « Comment cuire un œuf à la coque ? ».

Exemple : l'algorithme de l'œuf à la coque

Construire un algorithme, c'est avant tout analyser l'énoncé du problème afin de définir l'ensemble des objets à manipuler pour obtenir un résultat.

Définition des objets manipulés

Analysons l'énoncé suivant :

```
Comment cuire un œuf à la coque ?
```

Chaque mot a son importance. Ainsi, « à la coque » est aussi important que « œuf ». Le terme « à la coque » implique que l'on doit pouvoir mesurer le temps avec précision.

Notons que tous les ingrédients et ustensiles nécessaires ne sont pas cités dans l'énoncé. En particulier, nous ne savons pas si nous disposons d'une plaque de feu au gaz ou à l'électricité. Pour résoudre notre problème, nous devons prendre certaines décisions. Ces dernières vont avoir une influence sur l'allure générale de notre algorithme.

Supposons que pour cuire notre œuf nous soyons en possession des ustensiles et ingrédients suivants :

```
casserole
plaque électrique
```

```
eau
œuf
coquetier
minuteur
électricité
table
cuillère
```

En fixant la liste des ingrédients et des ustensiles, nous définissons un environnement, une base de travail. Nous sommes ainsi en mesure d'établir une liste de toutes les actions à mener pour résoudre le problème, et de construire la marche à suivre permettant de cuire notre œuf.

Liste des opérations

```
Verser l'eau dans la casserole, faire bouillir l'eau.
Prendre la casserole, l'œuf, de l'eau, le minuteur, le coquetier, la cuillère.
Allumer ou éteindre la plaque électrique.
Attendre que le minuteur sonne.
Mettre le minuteur sur 3 minutes.
Poser la casserole sur la plaque, le coquetier, le minuteur sur la table, l'œuf
dans la casserole, l'œuf dans le coquetier.
```

Cette énumération est une description de toutes les actions nécessaires à la cuisson d'un œuf.

Chaque action est un fragment du problème donné et ne peut plus être découpée. Chaque action est élémentaire par rapport à l'environnement que nous nous sommes donné.

En définissant l'ensemble des actions possibles, nous créons un langage minimal qui nous permet de cuire l'œuf. Ce langage est composé de verbes (`Prendre`, `Poser`, `Verser`, `Faire`, `Attendre`…) et d'objets (`Œuf`, `Eau`, `Casserole`, `Coquetier`…).

La taille du langage, c'est-à-dire le nombre de mots qu'il renferme, est déterminée par l'environnement. Pour cet exemple, nous avons, en précisant les hypothèses, volontairement choisi un environnement restreint. Nous aurions pu décrire des tâches comme « prendre un contrat d'accès à l'énergie électrique » ou « élever une poule pondeuse », mais elles ne sont pas utiles à notre objectif pédagogique.

Question

Quelle serait la liste des opérations si l'on décidait de faire un œuf poché ?

Réponse

Les opérations seraient :

```
Prendre du sel, du vinaigre, une assiette.
Verser le sel, le vinaigre dans l'eau.
Casser l'œuf et le verser dans l'eau.
Retirer l'œuf avec la cuillère.
```

Il est inutile de prendre un coquetier.

Ordonner la liste des opérations

Telle que nous l'avons décrite, la liste des opérations ne nous permet pas encore de faire cuire notre œuf. En suivant cette liste, tout y est, mais dans le désordre. Pour réaliser cette fameuse recette, nous devons ordonner la liste.

```
 1. Prendre une casserole.
 2. Verser l'eau du robinet dans la casserole.
 3. Poser la casserole sur la plaque électrique.
 4. Allumer la plaque électrique.
 5. Faire bouillir l'eau.
 6. Prendre l'œuf.
 7. Placer l'œuf dans la casserole.
 8. Prendre le minuteur.
 9. Mettre le minuteur sur 3 minutes.
10. Prendre un coquetier.
11. Poser le coquetier sur la table.
12. Attendre que le minuteur sonne.
13. Éteindre la plaque électrique.
14. Prendre une cuillère.
15. Retirer l'œuf de la casserole à l'aide de la cuillère.
16. Poser l'œuf dans le coquetier.
```

L'exécution de l'ensemble ordonné de ces tâches nous permet maintenant d'obtenir un œuf à la coque.

Remarque

L'ordre d'exécution de cette marche à suivre est important. En effet, si l'utilisateur réalise l'opération 12 (Attendre que le minuteur sonne) avant l'opération 9 (Mettre le minuteur sur 3 minutes), le résultat est sensiblement différent. La marche à suivre ainsi désordonnée risque d'empêcher la bonne cuisson de notre œuf.

Cet exemple tiré de la vie courante montre que pour résoudre un problème, il est essentiel de définir les objets utilisés, puis de trouver la suite logique de tous les ordres nécessaires à la résolution dudit problème.

Question

Où placer les opérations supplémentaires, dans la liste ordonnée, pour faire un œuf poché ?

Réponse

Les opérations s'insèrent dans la liste précédente de la façon suivante :

Entre les lignes 3 et 4,

```
Prendre le sel et le verser dans l'eau.
Prendre le vinaigre et le verser dans l'eau.
```

remplacer la ligne 7 par :

```
Casser l'œuf et le verser dans l'eau.
```

remplacer les lignes 10 et 11 par :

```
Prendre une assiette.
Poser l'assiette sur la table.
```

remplacer la ligne 16 par :

```
Poser l'œuf dans l'assiette.
```

Vers une méthode

La tâche consistant à décrire comment résoudre un problème n'est pas simple. Elle dépend en partie du niveau de difficulté du problème et réclame un savoir-faire : la façon de procéder pour découper un problème en actions élémentaires.

Pour aborder dans les meilleures conditions possibles la tâche difficile d'élaboration d'un algorithme, nous devons tout d'abord :

- déterminer les objets utiles à la résolution du problème ;
- construire et ordonner la liste de toutes les actions nécessaires à cette résolution.

Pour cela, il est nécessaire :

- d'analyser en détail la tâche à résoudre ;
- de fractionner le problème en actions distinctes et élémentaires.

Ce fractionnement est réalisé en tenant compte du choix des hypothèses de travail. Ces hypothèses imposent un ensemble de contraintes qui permettent de savoir si l'action décrite est élémentaire et peut ne plus être découpée.

Cela fait, nous avons construit un algorithme.

Et du point de vue de l'objet…

La programmation et par conséquent l'élaboration des algorithmes sous-jacents, s'effectuent aujourd'hui en mode objet.

Il s'agit toujours de construire des algorithmes et d'élaborer des marches à suivre, mais avec pour principe fondamental d'associer les actions (décrites dans la liste des opérations) aux objets (définis dans la liste des objets manipulés) de façon beaucoup plus stricte qu'en simple programmation.

Ainsi, une action est définie pour un type d'objet.

Reprenons par exemple le `minuteur` dans la liste des objets utilisés pour cuire notre œuf à la coque. En programmation objet, la liste des opérations concernant le `minuteur` peut s'écrire :

```
Minuteur_prendre
Minuteur_initialiser
Minuteur_sonner
```

Les termes `prendre`, `initialiser` et `sonner` représentent des blocs d'instructions qui décrivent comment réaliser l'action demandée. Ces actions sont accomplies uniquement si elles sont appliquées à l'objet `minuteur`. Ainsi, par exemple, l'instruction `œuf_sonner` ne peut être une instruction valide, car l'action `sonner` n'est pas définie pour l'objet `œuf`.

Grâce à ce principe, associer des actions à des objets, la programmation objet garantit l'exactitude du résultat fourni, en réalisant des traitements adaptés aux objets.

Passer de l'algorithme au programme

Pour construire un algorithme, nous avons défini des hypothèses de travail, c'est-à-dire supposé une base de connaissances minimales nécessaires à la résolution du problème. Ainsi, le fait de prendre l'hypothèse d'avoir une plaque électrique nous autorise à ne pas décrire l'ensemble des tâches consistant à allumer le gaz avec une allumette. C'est donc la connaissance de l'environnement de travail qui détermine en grande partie la construction de l'algorithme.

Pour passer de l'algorithme au programme, le choix de l'environnement de travail n'est plus de notre ressort. Jusqu'à présent, nous avons supposé que l'exécutant était humain. Maintenant, notre exécutant est l'ordinateur. Pour écrire un programme, nous devons savoir ce dont est capable un ordinateur et connaître son fonctionnement de façon à établir les connaissances et capacités de cet exécutant.

Qu'est-ce qu'un ordinateur ?

Notre intention n'est pas de décrire en détail le fonctionnement de l'ordinateur et de ses composants, mais d'en donner une image simplifiée.

Pour tenter de comprendre comment travaille l'ordinateur et, surtout, comment il se programme, nous allons schématiser à l'extrême ses mécanismes de fonctionnement.

Un ordinateur est composé de deux parties distinctes : la mémoire centrale et l'unité centrale.

La mémoire centrale permet de mémoriser toutes les informations nécessaires à l'exécution d'un programme. Ces informations correspondent à des « données » ou à des ordres à exécuter (« instructions »). Les ordres placés en mémoire sont effectués par l'unité centrale, la partie active de l'ordinateur.

Lorsqu'un ordinateur exécute un programme, son travail consiste en grande partie à gérer la mémoire, soit pour y lire une instruction, soit pour y stocker une information.

En ce sens, nous pouvons voir l'ordinateur comme un robot qui sait agir en fonction des ordres qui lui sont fournis. Ces actions, en nombre limité, sont décrites ci-dessous.

Déposer ou lire une information dans une case mémoire

La mémoire est formée d'éléments, ou cases, qui possèdent chacune un numéro (une adresse). Chaque case mémoire est en quelque sorte une boîte aux lettres pouvant contenir une information (une lettre). Pour y déposer cette information, l'ordinateur (le facteur) doit connaître l'adresse de la boîte. Lorsque le robot place une information dans une case mémoire, il mémorise l'adresse où celle-ci se situe afin de retrouver l'information le moment venu.

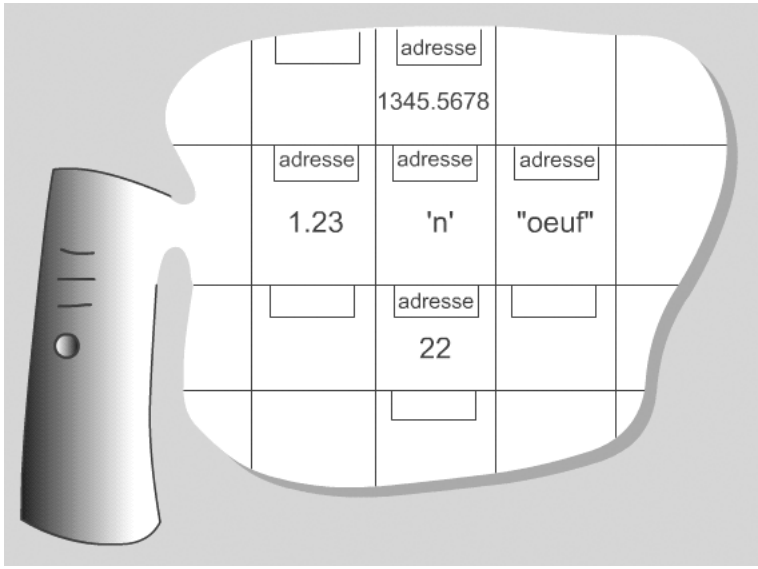

Figure I-1

La mémoire de l'ordinateur est composée de cases possédant une adresse et pouvant contenir à tout moment une valeur.

Le robot sait déposer une information dans une case, mais il ne sait pas la retirer (au sens de prendre un courrier déposé dans une boîte aux lettres). Lorsque le robot prend l'information déposée dans une case mémoire, il ne fait que la lire. En aucun cas il ne la retire ni ne l'efface. L'information lue reste toujours dans la case mémoire.

> **Remarque**
>
> Pour effacer une information d'une case mémoire, il est nécessaire de placer une nouvelle information dans cette même case. Ainsi, la nouvelle donnée remplace l'ancienne et l'information précédente est détruite.

Exécuter des opérations simples telles que l'addition ou la soustraction

Le robot lit et exécute les opérations dans l'ordre où elles lui sont fournies. Pour faire une addition, il va chercher les valeurs à additionner dans les cases mémoire appropriées (stockées, par exemple, aux adresses `a` et `b`) et réalise ensuite l'opération demandée. Il enregistre alors le résultat de cette opération dans une case d'adresse `c`. De telles opérations sont décrites à l'aide d'ordres, appelés aussi « instructions ».

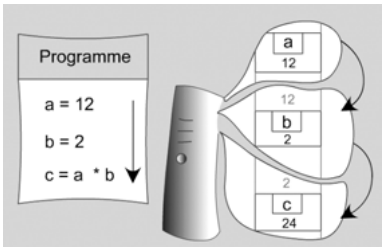

Figure I-2

Le programme exécute les instructions dans l'ordre de leur apparition.

Comparer des valeurs

Le robot est capable de comparer deux valeurs entre elles pour déterminer si l'une est plus grande, plus petite, égale ou différente de l'autre valeur. Grâce à la comparaison, le robot est capable de tester une condition et d'exécuter un ordre plutôt qu'un autre, en fonction du résultat du test.

La réalisation d'une comparaison ou d'un test fait que le robot ne peut plus exécuter les instructions dans leur ordre d'apparition. En effet, suivant le résultat du test, il doit rompre l'ordre de la marche à suivre, en sautant une ou plusieurs instructions. C'est pourquoi il existe des instructions particulières dites de « branchement ». Grâce à ce type d'instructions, le robot est à même non seulement de sauter des ordres, mais aussi de revenir à un ensemble d'opérations afin de les répéter.

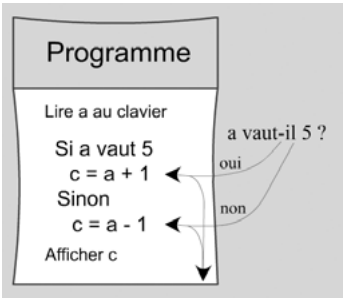

Figure I-3

Suivant le résultat du test, l'ordinateur exécute l'une ou l'autre des instructions en sautant celle qu'il ne doit pas exécuter.