# PARTIE 1

# Apprendre à programmer

# 1

# Qu'est-ce que programmer ?

L'ambition de ce livre est de démontrer que la programmation, abordée en douceur et avec pédagogie, n'est pas l'apanage des professionnels de l'informatique ; en effet, l'utilisateur lambda, s'il maîtrise les bases de la logique, peut apprendre aisément à programmer. Cette entreprise est à la portée de tous et cet ouvrage prétend démythifier la programmation, en montrant tout d'abord que cette discipline de l'informatique repose sur des techniques que chacun utilise dans la vie courante. Cela signifie que, comme Monsieur Jourdain faisait de la prose sans le savoir, vous avez déjà programmé, même si vous l'ignorez.

Nous définirons tout d'abord la programmation comme l'art d'écrire des programmes et nous dirons qu'un programme est une suite d'instructions. Le *Grand Robert* donne une définition plus complète que je vous livre : « *Ensemble ordonné des opérations nécessaires et suffisantes pour obtenir un résultat ; dispositif permettant à un mécanisme d'effectuer ces opérations*. »

Cette définition introduit la notion importante de résultat ; on programme toujours un ordinateur pour aboutir à un résultat. Nous reviendrons plus loin sur cet aspect non négligeable de la programmation.

On peut donc dire que lorsque vous écrivez une suite d'instructions, vous rédigez un programme. En fait, la réalisation en séquence d'une liste d'ordres est une opération assez banale dans la vie quotidienne et quand, par exemple, on réalise une recette de cuisine, on exécute un programme. Voici une recette facile que les adeptes du régime Dukan ne renieront certainement pas :

**Rillettes aux deux saumons**

Découper grossièrement en petits dés un pavé de saumon cru de 200 grammes et faites-le mariner au réfrigérateur pendant 4 heures dans de l'aneth, du sel, du poivre et le jus d'un citron vert. Mélanger la préparation toutes les heures.
Une fois le saumon cru mariné, ajouter 200 grammes de saumon fumé et mixer le tout. Rajouter 400 grammes de fromage blanc à 0 % de matière grasse ainsi qu'une cuillère à soupe de moutarde à l'ancienne et un peu de vinaigre balsamique.
Afin de rendre la préparation plus ferme, rajouter 10 cuillers à soupe de son d'avoine et bien mélanger, puis mettre au réfrigérateur pendant deux heures.
Rectifier l'assaisonnement en cas de besoin et rajouter éventuellement de la ciboulette, du persil et des câpres.
Servir sur du pain grillé ou des galettes aux sons de blé et d'avoine.

Dans cette recette de cuisine qui est à la portée de tous, on trouve en fait une bonne partie des concepts de la programmation que nous étudierons tout au long de cet ouvrage, comme les boucles, les tests conditionnels et les fonctions.

Si vous n'êtes pas très porté sur la gastronomie et que cet exemple ne vous dit pas grand-chose, vous avez sans doute déjà réalisé le montage d'un meuble en kit ; cette opération s'apparente également à la réalisation d'un programme informatique. Si vous commencez à réfléchir à certaines opérations de la vie quotidienne, vous vous rendrez alors compte qu'il existe de nombreuses activités où l'on doit reproduire en séquence toute une série d'actions afin d'aboutir à un résultat. Prendre son petit-déjeuner le matin ou bien se laver les dents sont en général des activités qui sont parfaitement codifiées et que vous accomplissez tous les jours sans vous poser de questions. Pourtant, au sens informatique du terme, il s'agit de programmes que vous exécutez. Programmer consiste à écrire le scénario complet de ces activités pour arriver à un résultat toujours identique ; dans le cas du petit-déjeuner, le but est d'ingérer des aliments qui apporteront suffisamment de calories pour vous permettre de tenir le coup jusqu'au repas de midi. Exécuter un programme consiste à effectuer les unes après les autres les différentes instructions d'un scénario qui bien entendu n'a pas besoin d'être écrit dans la vie courante : prendre le tube de dentifrice, ouvrir le tube, étaler la pâte sur la brosse à dents, refermer le tube, etc.

Grâce à ces exemples extraits de la vie quotidienne, on constate facilement que la logique et les concepts de la programmation nous sont en fait très familiers. Il n'y a donc pas lieu de redouter la programmation informatique car nous en possédons la plupart de ces mécanismes ; les seules choses qui vont changer sont le but que l'on va assigner au programme et le langage qui va permettre de décrire le déroulement des opérations à exécuter.

## PLUSIEURS NIVEAUX DE PROGRAMMATION

De la même manière qu'il existe des recettes de cuisine plus ou moins compliquées, il existe plusieurs niveaux de programmation. On peut considérer que le premier niveau de programmation dans Office consiste ni plus, ni moins, à paramétrer le logiciel afin qu'il réponde à nos exigences particulières. Ainsi, le simple fait de renseigner la boîte de dialogue des options de Word est une programmation basique dans la mesure où l'on va donner des instructions à Word pour qu'il se comporte de la manière souhaitée (par exemple, afficher les codes de champ). De la même manière, la réorganisation du ruban est aussi une forme élémentaire de programmation.

Le deuxième niveau est l'automatisation de certaines tâches répétitives grâce à la sauvegarde des opérations accomplies les unes à la suite des autres : on parle alors de **macro-commandes** (ou macros). Il existe certains logiciels (notamment Word et Excel) qui permettent d'enregistrer la séquence des opérations que vous êtes en train de réaliser et qui vous autorisent ensuite à rejouer cette séquence quand vous le désirez. C'est un peu le principe du magnétoscope : vous enregistrez et vous rejouez autant de fois que vous le voulez et quand vous le voulez.

Le troisième niveau est l'écriture de fonctions qui sont absentes du logiciel que vous utilisez, que ce soit le système d'exploitation ou bien un des logiciels de la suite Office. Imaginons que vous ayez souvent besoin dans Excel de convertir des valeurs exprimées en minutes en valeurs exprimées en heures ; ainsi la valeur « 230 » devra être convertie en « 3 heures et 50 minutes ». À ma connaissance, une telle fonction n'existe pas dans Excel et vous pouvez, à l'aide du langage de programmation d'Office, écrire votre propre fonction de conversion et faire en sorte que votre programme devienne une nouvelle fonction intégrée d'Excel.

Le dernier niveau est l'écriture de programmes complets prenant en charge une tâche complexe, par exemple un logiciel de facturation. Le programme prend en compte tous les aspects d'une application : l'interface utilisateur (les boîtes de dialogue et les formulaires de saisie), les calculs et les impressions.

Un programme consiste donc en une séquence d'instructions nécessaires pour atteindre un but. Avant d'écrire un programme, il faut toujours déterminer précisément le but à atteindre et chacun comprendra que plus l'objectif est complexe, plus le programme sera long et difficile à écrire.

## LES LANGAGES DE PROGRAMMATION

La recette de cuisine citée plus haut est rédigée en langage naturel alors que les programmes informatiques s'écrivent à l'aide de langages de programmation. De la même manière que les langues vivantes sont censées obéir à des règles de grammaire, les langages de programmation suivent des règles que l'on nomme **syntaxe**. Alors que les langues naturelles tolèrent assez bien les approximations (tout le monde comprend la phrase « J'ai même pas eu peur » malgré l'absence de la négation), les langages informatiques sont beaucoup plus puristes et pointilleux, si bien que la moindre omission d'une virgule, d'une parenthèse ou bien d'un point sera immédiatement sanctionnée. Le caractère strict de la syntaxe d'un langage informatique est parfois mal vécu par les apprentis programmeurs ; il faut bien comprendre que l'ordinateur, à la différence d'un être humain, ne peut pas interpréter les mots qui manquent et les phrases mal construites. L'architecture binaire d'un ordinateur a pour conséquence qu'un programme est syntaxiquement correct ou incorrect et qu'il ne peut pas y avoir de juste milieu. Un programme peut donc planter, c'est-à-dire s'arrêter brutalement, parce que vous avez oublié un point-virgule dans le code.

**Définition**

On appelle **code** ou **code source**, voire **source**, l'ensemble des lignes d'un programme et **encoder** ou **coder** le fait de transcrire les actions à exécuter dans un langage informatique.

## LA SYNTAXE

Le code d'un programme est composé de phrases élémentaires appelées lignes d'instruction. Chaque ligne d'instruction doit exécuter une action comme l'affichage d'un message à l'écran, l'addition de deux nombres, la lecture d'une valeur stockée dans un fichier, etc. Chaque langage de programmation possède sa propre syntaxe, c'est-à-dire ses propres règles d'écriture. Les lignes d'un programme doivent être écrites avec le vocabulaire du langage de programmation qui comprend un nombre de mots fini. Comme dans une langue naturelle, il existe plusieurs catégories de mots (verbe, adjectif, conjonction de coordination, etc.) dans un langage de programmation et nous apprendrons, au fur et à mesure de notre progression, ces différents types de mots.

Comme un énoncé humain, une instruction peut être ambiguë et il convient à tout prix d'éviter les ambiguïtés. Ainsi, le résultat de l'instruction qui effectue le calcul suivant :

```
x = 2 + 3 * 4
```

paraît incertain car on ne sait pas si *x* vaut 20 ou 14. La simple utilisation de parenthèses lèvera, dans le cas présent, l'ambiguïté.

**Remarque**

En réalité, la plupart des langages de programmation considéreront qu'il n'y a pas d'ambiguïté dans cette formule de calcul car l'opérateur de la multiplication est prioritaire sur celui de l'addition. Les opérateurs mathématiques (+, -, * et /) ont un degré de priorité les uns par rapport aux autres qui détermine l'ordre dans lequel les opérations mathématiques sont effectuées.

## LES PHASES DE CONCEPTION D'UN PROGRAMME

Quel que soit le langage employé pour écrire un programme, il existe une méthodologie pour le rédiger. On a l'habitude de décomposer l'écriture d'un programme en différentes phases.

### La phase d'étude préalable

S'il fallait résumer cette première étape par une maxime, nous proposerions : « réfléchir avant d'agir ! ». En effet, avant d'écrire un programme

quelconque, la première des choses à faire est d'éteindre son ordinateur et de réfléchir. On peut notamment commencer par se poser les questions suivantes :

- Quel est l'objectif de ce programme ?
- N'est-il pas plus rapide de réaliser cet objectif manuellement ?
- Cet objectif a-t-il réellement un intérêt ?
- Ce programme n'existe-t-il pas déjà sous une autre forme ?
- Ce programme est-il réalisable ?
- La réalisation de ce programme n'est-elle pas trop coûteuse ?

Bien évidemment, il existe de nombreux cas où vous pourrez écrire un programme sans vous poser toutes ces questions. Ainsi, quand vous voudrez rédiger un programme très simple pour automatiser une tâche précise qui n'est pas complexe, vous pourrez foncer bille en tête. En revanche, dès que le projet de programmation devient un peu plus ambitieux, il vaut vraiment mieux se poser des questions avant de programmer. Cette manière de faire s'apparente (ou devrait s'apparenter) à la pratique des informaticiens professionnels. En tant qu'amateur, vous pensez peut-être pouvoir vous dispenser de toute cette rigueur qui est l'apanage du professionnel, mais vous auriez tort d'agir de la sorte. On peut programmer en dilettante tout en adoptant une démarche professionnelle ; cela n'est pas contradictoire ! En fait, la programmation est une discipline exigeante et si l'on ne respecte pas un minimum les règles du jeu, on risque de ne pas arriver au but que l'on s'était assigné, ce qui engendrera déconvenues et frustrations. De très nombreux projets informatiques ne sont pas menés jusqu'au bout car on a négligé la phase de définition de l'objectif du logiciel. Si cette description n'est pas assez complète, tout l'édifice risque d'être compromis. Ne perdez jamais de vue que l'on ne programme pas pour programmer, mais toujours pour atteindre un but. Quand un architecte dessine les plans d'une maison, il doit avoir une idée précise de ce que souhaite son client.

## La phase d'analyse

Une fois que l'on a l'assurance que le projet de programmation est réalisable, il faut réfléchir à la structuration du programme. L'informatique étant la science du traitement automatisé de l'information, un programme n'est jamais qu'un processus de transformation d'informations. Il convient donc d'inventorier toutes les informations dont le programme a besoin au départ et toutes les informations dont il aura besoin en sor-

tie. Quand on possède toutes ces données, il faut décrire les algorithmes qui permettront de transformer les informations disponibles en entrée en résultats.

**Définition**

Un **algorithme** est l'ensemble des règles opératoires qui permettent d'effectuer un traitement de données ; ce procédé décrit formellement toutes les étapes d'un calcul qui doit fonctionner dans tous les cas de figure.

Par exemple, voici l'algorithme pour trouver si un nombre entier est pair :

- Diviser le nombre entier par 2,
- Si le reste de la division est 0, le nombre est pair,
- Sinon, le nombre est impair.

On peut alors décrire tout le déroulement du programme dans un langage quasi naturel que l'on appellera pseudo-code. Voici un exemple de pseudo-code qui permet d'appliquer un tarif réduit pour les mineurs :

- Demander à l'utilisateur sa date de naissance,
- Si l'utilisateur a moins de 18 ans,
- Diviser le prix par deux,
- Sinon appliquer le prix normal.

## La phase d'encodage

Une fois que l'analyse est terminée, il faut transcrire le pseudo-code dans un langage de programmation. Les phases d'étude et d'analyse sont indépendantes de tout langage de programmation et le choix de ce dernier peut se faire au moment de l'encodage. Plus la phase d'analyse a été poussée, plus l'encodage sera simple. La plupart des problèmes de programmation proviennent d'une analyse trop succincte, voire d'une absence totale d'analyse.

## La phase de test

Quand l'encodage est achevé, il faut tester le programme car il est excessivement rare qu'un programme, sauf s'il est très court et extrêmement simple, fonctionne correctement du premier coup. Les causes d'erreur

sont multiples et les tests permettent de les mettre en évidence ; il faut alors revenir en arrière et retourner, en fonction de la gravité de l'erreur, à la phase d'analyse (erreur de conception) ou d'encodage (erreur de programmation).

### La phase de production

Une fois que le programme paraît exempt d'erreurs (ce n'est malheureusement souvent qu'une illusion…), on peut envisager de le diffuser auprès des utilisateurs.

Le cycle de vie du logiciel n'est pas pour autant terminé car il est fort probable que certains utilisateurs trouvent des bugs (erreurs de programmation) qui n'auront pas été détectés lors des phases de tests ou bien que d'autres utilisateurs demandent au programmeur des améliorations ou de nouvelles fonctionnalités. Il faudra alors se relancer dans une analyse, voire repartir de zéro si les modifications souhaitées sont trop importantes…

## VBA : UN LANGAGE DE PROGRAMMATION POUR LES APPLICATIONS

VBA est l'acronyme de Visual Basic pour Applications et vous rencontrerez parfois la dénomination Visual Basic Edition Application qui est tombée en désuétude. Il s'agit donc d'une version de Visual Basic pour les applications. Le langage de programmation **Basic** est un langage assez ancien qui a été créé en 1965 ; langage d'initiation (Basic signifie *Beginner's All-purpose Symbolic Instruction Code*), il a connu d'innombrables versions sur la plupart des systèmes d'exploitation. Pour Bill Gates, il s'agit pourtant d'un langage fétiche car c'est le premier programme qu'il a écrit et commercialisé avec son ami Paul Allen. Il s'agissait à l'époque d'une version de Basic pour un ordinateur baptisé Altair. Lorsque nos deux compères créèrent Microsoft et proposèrent leur système d'exploitation à IBM, une version du langage Basic était bien évidemment proposée dans le package. Chacun connaît la suite de l'histoire…

Avec l'avènement de Windows, les interfaces utilisateur sont devenues graphiques et Microsoft se devait de faire évoluer son Basic : c'est ainsi que Microsoft Basic est devenu Visual Basic. Simple et visuelle, cette nouvelle version du langage obtint un succès formidable

et aujourd'hui, Visual Basic est un langage de programmation encore très utilisé. Mais le rêve de Bill Gates était véritablement d'imposer ce langage à tous les produits que commercialisait Microsoft. On a donc vu apparaître en 1993 une version minimale de Visual Basic dans Excel et cette version fut appelée VBA. Puis ce fut le tour de Project et d'Access d'accueillir VBA ; dans le cas d'Access, VBA venait remplacer Access Basic. En 1996, sortit la version 4 de Visual Basic et VBA remplaça Word Basic. Une année plus tard, la version 5 de Visual Basic vit le jour et chaque application de la suite Office 97 (à l'exception d'Outlook) incorporait désormais une version de VBA, même si de légères différences entre les applications subsistaient encore. En 1998, Microsoft livra Visual Basic 6 et c'est cette dernière version qui est présente dans Office 2000, Office XP, Office 2003 et Office 2007, Office 2010 accueillant la version 7 de VBA.

## Différences entre Visual Basic et VBA

La principale différence entre Visual Basic et VBA réside dans le fait que VBA a besoin d'une application hôte pour pouvoir exécuter ses programmes. Les applications hôtes de VBA sont essentiellement les applications de la suite Office, mais d'autres programmes, comme Autocad, peuvent être programmés à l'aide de VBA. Si vous écrivez une macro en VBA pour Word, vous devez absolument posséder Word pour faire tourner votre programme. En revanche, si vous écrivez un programme en Visual Basic, vous pouvez le compiler afin de produire un fichier exécutable autonome qui pourra être lancé sur un ordinateur qui ne dispose pas de Visual Basic. À cette différence près, les deux langages sont extrêmement proches et il est particulièrement aisé de passer de l'un à l'autre.

**Définition**

Quand un programme est **compilé** (à l'aide d'un compilateur), son code source (les instructions du programme) est transformé en code machine et on obtient au final un programme exécutable (avec une extension .EXE). Les programmes écrits en VBA ne peuvent pas être compilés ; on dit qu'ils sont **interprétés** (à l'aide d'un interpréteur). Chaque application Office possède un interpréteur VBA qui permet d'exécuter les programmes écrits en VBA. Les programmes interprétés s'exécutent moins rapidement que les programmes compilés.

## CONCLUSION

Un programme doit avoir un but bien déterminé et la programmation consistera à écrire les instructions permettant de réaliser un objectif. Avant de commencer à programmer, il faut bien réfléchir à la structure du programme et inventorier les informations qui sont manipulées par le programme. Apprendre à programmer, c'est apprendre un langage de programmation qui est composé d'un vocabulaire (une liste de mots finie dont on peut consulter chaque définition dans l'aide en ligne) et d'une syntaxe (la manière d'agencer les mots). Programmer n'est pas difficile si l'on a l'esprit un tant soit peu logique et si l'on respecte rigoureusement la syntaxe du langage de programmation que l'on utilise, car la moindre erreur de syntaxe peut bloquer le programme.

VBA est le langage de programmation d'Office.