DOMINIQUE MANIEZ

Gagnez du temps avec Excel et Word

Programmation VBA pour Mac et PC

DUNOD

Direction artistique : Élisabeth Hébert

11 rue Paul Bert, 92240 Malakoff
www.dunod.com

ISBN 978-2-10-081092-5

Table des matières

Avant-propos

Il y a tout juste vingt ans, j'ai écrit le premier livre en français qui traitait de la programmation VBA (Visual Basic pour Application) des cinq applications de la suite Office, à savoir Word, Excel, Access, Outlook et PowerPoint. À l'époque, je voulais faire découvrir la puissance de ce langage de programmation qui permettait de décupler la productivité de ces outils bureautiques qui nous rendaient déjà de grands services et nous évitaient des tâches fastidieuses (que l'on songe, par exemple, au publipostage). Le succès de cette première édition a dépassé toutes mes espérances et j'étais bien entendu ravi de partager avec mes lecteurs les joies de la programmation de macros qui nous faisaient gagner un temps précieux. Je n'ai cependant jamais compris le dédain, voire le mépris, des informaticiens professionnels à l'égard de ce langage auquel ils reprochaient de nombreux défauts. Ayant programmé dans de nombreux langages, je veux bien reconnaître que VBA a des lacunes, mais quand on a à sa disposition un langage qui s'apprend sans peine (n'oublions pas que Basic est un acronyme dont le B est l'initiale de *Beginners* qui signifie « débutants » en anglais) et qui facilite la vie, il faut savoir être pragmatique et profiter de l'aubaine. En tous les cas, telle est ma philosophie.

ÉCONOMISEZ VOTRE TEMPS ET VOTRE PEINE

Après vingt ans de programmation de macros, je peux dire que VBA n'a pas pris une ride et qu'il m'économise toujours autant de temps et de peine. De nos jours, l'apprentissage du codage, notamment chez les jeunes, est souvent présenté comme une discipline intellectuelle permettant de se familiariser avec la pensée informatique. Former les esprits est une excellente chose, mais on a parfois un peu tendance à oublier qu'un programme permet avant tout de gagner du temps et de faire l'économie de tâches parfois très fastidieuses. La fonction première des ordinateurs a été d'accroître la productivité, et la programmation, même à un niveau modeste, va vous permettre de gagner un temps précieux, surtout si vous accomplissez des tâches répétitives. En effet, l'automatisation des opérations manuelles accomplies quotidiennement va augmenter votre productivité, parfois dans des proportions que vous n'imaginez même pas. Vous

gagnez ainsi sur les deux tableaux : vous économisez beaucoup de temps tout en vous affranchissant des tâches pénibles.

En une vingtaine d'années de programmation VBA, je n'ai jamais calculé le temps que VBA m'a fait gagner, mais cela se chiffre sans doute en mois de travail. Vous allez penser que j'exagère, mais je vais vous donner un exemple simple afin d'illustrer mon propos. Récemment, à l'issue de la traduction d'un ouvrage passionnant sur le refactoring[1], mon éditeur m'a signalé que pour l'édition électronique du livre il était nécessaire de créer des signets qui se transformeraient en liens hypertexte lors de la génération du fichier ePub (le format des livres électroniques). Cette manipulation est relativement simple à effectuer manuellement, mais elle est minutieuse, et fastidieuse quand elle se répète sur un volume de plus quatre cents pages. J'ai d'abord utilisé une macro pour comptabiliser le nombre de liens à générer et je me suis aperçu qu'il y en avait plus de cinq cents… Si j'avais dû accomplir cette tâche manuellement, j'en aurais eu pour une journée complète de travail, avec un risque d'erreur important car le cerveau (en tous les cas le mien), quand on lui confie un travail fastidieux, décroche au bout d'un moment et produit en général des résultats hasardeux. J'ai mis un peu moins d'une heure à écrire et à tester ma macro et je considère donc que j'ai économisé sept heures de travail. Des exemples tels que celui-ci, j'en ai des centaines, dans des domaines très variés, et mon ambition est de partager avec vous cette efficacité qu'apportent les macros afin que vous consacriez le temps ainsi préservé à d'autres activités. Il faut aussi noter que les gains de productivité apparaissent très vite et que le retour sur investissement est par conséquent rapide ; il n'est donc pas nécessaire d'être devenu un expert en programmation VBA pour commencer à voir les effets bénéfiques de ses propres macros. Pour le dire autrement, ce sont souvent les macros relativement courtes qui font gagner le plus de temps.

À QUI S'ADRESSE CE LIVRE

Cet ouvrage est destiné à celles et ceux qui utilisent Word ou Excel de manière intensive, la plupart du temps dans un cadre professionnel, et qui souhaitent optimiser leur usage de ces applications, afin de gagner encore plus de temps. Pour apprendre à programmer en VBA, il est nécessaire de bien connaître le fonctionnement de la suite Office. Par exemple, il paraît difficile de vouloir programmer Excel si vous n'utilisez pas de formules de calcul ni de fonctions intégrées, comme SOMME(), MIN, ou bien encore SI(). Pour autant, aucune connaissance préalable des concepts de programmation n'est requise et il n'est nul besoin d'avoir fait de grandes études de mathématiques, la simple maîtrise des quatre opérations arithmétiques de base étant largement suffisante. Cet ouvrage a vraiment été conçu pour des personnes qui n'ont aucune expérience de la programmation et c'est un peu ma marque de fabrique, puisque j'ai déjà écrit plusieurs livres consacrés à l'apprentissage de la programmation pour les gens qui n'y connaissent rien et qui n'ont pas de culture scientifique[2]. En vingt ans, ma méthode pédagogique a évolué et c'est la raison pour laquelle l'ouvrage que vous tenez entre les mains est différent de celui que j'ai rédigé en 2000. Ma conviction profonde est que la majorité des ouvrages de programmation sont faits pour les informaticiens et les personnes

1. https://www.dunod.com/sciences-techniques/refactoring-comment-ameliorer-code-existant
2. https://www.dunod.com/sciences-techniques/apprendre-programmer-en-10-semaines-chrono-une-methode-visuelle-pour-tous

qui maîtrisent déjà un certain formalisme, raison pour laquelle ils ne conviennent donc pas à celles et ceux qui s'initient au développement informatique et qui ne sont pas habitués à manipuler certaines abstractions. Par exemple, j'ai constaté qu'il vaut mieux commencer par étudier des exemples de programmes plutôt que la syntaxe exhaustive d'un langage de programmation. Prenons une comparaison avec l'apprentissage des langues vivantes : si l'on demandait à un élève de ne commencer à parler l'anglais qu'après avoir appris toutes les règles de grammaire ainsi qu'un vocabulaire de base conséquent, l'élève se découragerait vite et ne progresserait pas. C'est la raison pour laquelle on n'apprend pas les langues vivantes de cette façon et que l'on plonge les élèves dans un bain linguistique dès le départ, même s'ils ne comprennent pas la totalité des productions orales ou écrites auxquelles on les soumet. Cet apprentissage par imprégnation donne de bons résultats, mais il faut accepter de ne pas tout comprendre tout de suite et d'avoir, dans un premier temps, une vision partielle des choses. Ce livre a donc été conçu selon ces principes et les notions du langage VBA sont étudiées au fur et à mesure du déroulement de la progression de l'ouvrage, en présentant les exemples avant l'apprentissage de la syntaxe et du vocabulaire.

Après avoir lu cet ouvrage :

- vous aurez une bonne idée de ce qu'est la programmation ;
- vous maîtriserez les concepts de base de la programmation ;
- vous saurez écrire de petits programmes VBA pour Word et Excel ;
- vous pourrez apprendre un langage de programmation plus puissant.

LA FIN DU MICMAC POUR LA POMME

Ce livre est également différent des précédents car il fait la part belle aux versions d'Office pour Macintosh. Par le passé, quand je vantais les mérites de VBA, les personnes qui travaillaient avec un Mac avaient beaucoup de mal à adhérer à mon propos car elles ne pouvaient tout simplement pas mettre en pratique les techniques dont je faisais la démonstration dans mes livres. La version VBA pour Macintosh a mis du temps à apparaître et elle a connu de nombreuses évolutions (elle a même tout simplement disparu de la version d'Office 2008), mais depuis quelques années, l'environnement de développement sous Mac s'est stabilisé et propose une relativement bonne compatibilité avec les versions d'Office pour Windows. Cet ouvrage s'adresse donc à la fois aux utilisateurs de Word et d'Excel qui utilisent un Mac ou un PC. Toutes les différences entre les deux versions sont documentées et un chapitre de ce livre est même consacré à la programmation des macros pouvant s'exécuter indifféremment sur les deux plateformes.

QUELLE VERSION D'OFFICE ?

Comme je l'ai déjà évoqué, le développement des versions d'Office ne s'est pas fait au même rythme sur Mac et sur PC et il en va de même pour les versions de VBA qui sont intégrées aux applications comme Word et Excel. Les programmes de cet ouvrage ont été testés avec les versions de Word et Excel d'Office 2016 pour Mac et PC. Si vous avez une version supérieure, par exemple Office 2019, vous ne devriez avoir aucun problème pour exécuter les programmes contenus dans cet ouvrage car la version actuelle de VBA (la version 7.1) n'a pas évolué depuis Office 2013. De la même manière, si vous

travaillez avec Office 2013, vous ne devriez pas éprouver des difficultés puisqu'il s'agit de la même version de VBA. En revanche, si vous avez une version d'Office antérieure à 2013, il n'est pas garanti que tous les programmes fonctionneront.

Peu d'utilisateurs le savent, mais pour chaque version d'Office, il existe une version 32 bits et une version 64 bits. La plupart des utilisateurs installent la version 32 bits et les programmes de cet ouvrage sont basés sur cette version, mais j'aborderai à la fin du livre les spécificités de la version 64 bits.

WWW.VBAPOURTOUS.FR

Il n'existe pas de version électronique de ce livre, mais cet ouvrage est complété par un site Web qui propose des suppléments en ligne.

Ce site Web a d'abord pour vocation d'économiser du papier et de n'imprimer que le strict code nécessaire dans ce livre. En tant que lecteur d'ouvrages d'informatique, je me suis toujours interrogé sur la pertinence et sur la plus-value d'un ouvrage papier qui intègre des dizaines, voire des centaines de pages de listing que le lecteur lit rarement dans leur intégralité et dont il se contente de télécharger le code pour les utiliser. Il est entendu qu'un ouvrage de programmation doit donner des exemples de code, mais est-il nécessaire d'imprimer la totalité des programmes, surtout quand ils sont très longs ?

Dans ce livre, je ne reproduis des programmes complets que lorsqu'ils sont très courts et pour les programmes plus longs, ne sont imprimées que les parties du code qui sont commentées. La plupart des exemples utilisés sont donc incomplets, mais les extraits sélectionnés sont les plus intéressants et ceux sur lesquels il faut porter votre attention. Les codes complets sont par conséquent disponibles sur un site Web que vous trouverez à l'adresse :

www.vbapourtous.fr

Ce site Web a une deuxième ambition : instaurer un dialogue avec mes lecteurs. En effet, il est possible de poster un commentaire sur chaque exemple de code de cet ouvrage de telle sorte que vous pouvez poser une question si vous ne comprenez pas bien un exemple ; je me ferai alors un plaisir de vous répondre en précisant et en complétant mon propos. Si vous avez trouvé un autre exemple plus pertinent ou bien une variante intéressante de l'un des programmes que je propose, vous pourrez aussi, à votre tour, partager vos connaissances avec les autres lecteurs.

Ce site Web comprend également une série de documentations électroniques qui n'ont pas leur place dans un ouvrage papier, mais qui sont néanmoins importantes quand on apprend à programmer.

Importance des exemples de code

Je ne le répéterai jamais assez : on apprend à programmer en écrivant du code et en l'exécutant, mais aussi en étudiant le code de programmes écrits par d'autres. Dans ces conditions, si vous voulez progresser, il est impératif que vous téléchargiez l'intégralité du code des macros qui servent d'exemples dans cet ouvrage. L'accès au site www.vbapourtous.fr n'est donc pas facultatif...

1 Introduction à la programmation

POURQUOI APPRENDRE À PROGRAMMER

Si vous avez ce livre entre les mains, c'est que vous avez envie d'apprendre à programmer Word et Excel. J'ignore vos motivations profondes, mais j'aimerais au début de cet ouvrage vous montrer que les raisons qui peuvent pousser à se lancer dans cette aventure sont vraiment nombreuses et dépassent très largement le gain de productivité que j'ai évoqué dans l'avant-propos et qui est sans doute votre principale motivation. Mais avant toute chose, vous devez savoir que ce livre a été conçu pour tous ceux qui n'ont jamais écrit une seule ligne de programme informatique de leur vie et qu'il s'adresse vraiment aux débutants, les prérequis pour bénéficier de cet ouvrage étant minimalistes.

Aujourd'hui, tout le monde apprend à coder !

L'apprentissage du codage informatique est devenu obligatoire au collège depuis la rentrée 2016, et les élèves du cycle 4 (de la 5e à la 3e) reçoivent « *un enseignement d'informatique, dispensé à la fois dans le cadre des mathématiques et de la technologie. Celui-ci n'a pas pour objectif de former des élèves experts, mais de leur apporter des clés de décryptage d'un monde numérique en évolution constante. Il permet d'acquérir des méthodes qui construisent la pensée algorithmique et développe des compétences dans la représentation de l'information et de son traitement, la résolution de problèmes, le contrôle des résultats.* » À la rentrée 2019, un enseignement intitulé « sciences numériques et technologie » a été généralisé en classe de seconde pour tous les élèves de lycée. Apporter des clés pour décrypter un monde numérique en constante évolution est véritablement un objectif socialement et politiquement souhaitable.

Apprendre à programmer est très formateur pour l'esprit

Dans les années 1960, on vantait les mérites de l'apprentissage du latin dont la grammaire était censée faire des merveilles pour le développement de l'esprit. Même si cette discipline était sans doute plus un marqueur social important dans la reproduction des élites, il n'en reste pas moins que certaines matières sont extrêmement formatrices et la programmation en fait partie. Outre la logique qui est une part importante de la programmation, la recherche de solutions, leur formalisation ainsi que leur modélisation sont des éléments capitaux du codage qui met en œuvre des mécanismes cognitifs que l'on retrouvera d'ailleurs dans de nombreux aspects de la vie quotidienne. Il est par conséquent indubitable que l'apprentissage de la programmation a un effet bénéfique sur la formation de l'esprit.

Apprendre à coder permet de trouver du travail

Même si la finalité de cet ouvrage n'est pas de former des informaticiens professionnels, il est indéniable que l'on manque de codeurs en France et qu'avoir des compétences en ce domaine peut ouvrir des portes. Aujourd'hui, bon nombre de diplômés en sciences humaines de l'enseignement supérieur se reconvertissent avec succès dans l'informatique après une formation initiale qui n'a aucun rapport avec cette discipline. Il est également nécessaire de ne pas hypothéquer l'avenir et de nombreuses études prédisent qu'une bonne part des emplois qualifiés qui vont émerger dans les dix prochaines années n'existent pas encore, une partie non négligeable de ces emplois relevant du secteur de l'activité numérique.

Coder permet de mieux utiliser un ordinateur

Apprendre à programmer, c'est devenir acteur du processus informatique. Quand on programme, on est moins passif devant sa machine et on acquiert une meilleure connaissance du fonctionnement matériel et logiciel de l'ordinateur. Quand on veut comprendre la culture d'un pays étranger, il est nécessaire de connaître les rudiments de la langue parlée par les autochtones. Il en va de même pour l'informatique : vous devez apprendre un langage de programmation pour mieux comprendre la culture informatique qui, au fil du temps, a pénétré tous les secteurs de la société.

Il vaut mieux programmer qu'être programmé !

Aujourd'hui, l'informatisation de la société est poussée à l'extrême et c'est finalement un enjeu social et politique que de comprendre comment les programmes fonctionnent. Nous sommes un certain nombre à penser que la chose informatique ne doit pas être l'apanage des informaticiens professionnels parce qu'à l'heure actuelle l'informatique concerne tout le monde. On a fêté en 2018 les quarante ans de la loi *Informatique et libertés* et il apparaît de plus en plus urgent de faire respecter le premier article de la loi du 6 janvier 1978 dont l'énoncé semble aujourd'hui un idéal lointain :

« L'informatique doit être au service de chaque citoyen. Son développement doit s'opérer dans le cadre de la coopération internationale. Elle ne doit porter atteinte ni à l'identité humaine, ni aux droits de l'homme, ni à la vie privée, ni aux libertés individuelles ou publiques. »

À la lecture de cet argumentaire, on voit bien que l'apprentissage de la programmation procède à la fois d'enjeux culturels, intellectuels et citoyens.

NOTIONS ÉLÉMENTAIRES DE PROGRAMMATION

Tentons à présent de donner une définition de la programmation afin d'être certain que nous parlons tous de la même chose. Nous définirons tout d'abord la programmation comme l'art d'écrire des programmes. Bien entendu, le terme « programme » est ici pris au sens informatique où il a une signification bien précise et on peut, par exemple, reprendre la définition du Grand Robert :

« Ensemble ordonné des opérations nécessaires et suffisantes pour obtenir un résultat ».

Cette définition introduit la notion importante de **résultat** ; on ne programme pas pour le plaisir de programmer, mais on programme toujours pour aboutir à un résultat.

Un programmeur a un but, une idée bien précise du résultat qu'il veut obtenir et une partie importante des problèmes que l'on rencontre quand on programme vient du fait que l'on n'a pas défini clairement au départ le résultat auquel on veut parvenir.

L'autre élément important de la définition est que les opérations sont ordonnées. Cela peut paraître évident, mais il est préférable de le rappeler : l'ordre dans lequel les opérations sont effectuées est capital. Tous ceux qui cuisinent savent cela et il en va de même en informatique. Le fait d'exécuter des opérations dans un sens plutôt qu'un autre pourra, dans de très nombreux cas, faire échouer le programme.

On peut aussi déduire de cette définition qu'il n'existe peut-être pas un ensemble unique d'opérations pour arriver à un résultat. De la même manière qu'il existe en général plusieurs chemins pour aller d'un point A à un point B, il peut y avoir plusieurs solutions à un même problème informatique. Certains chemins sont plus courts et d'autres plus rapides pour arriver à la même destination. Certains programmes seront aussi plus rapides que d'autres pour arriver au même résultat, et la différence de temps d'exécution peut être importante entre deux programmes équivalents en termes de fonctionnalités. On privilégiera en général la version la plus rapide et l'efficacité d'un programme sera un des critères d'appréciation de sa qualité. Un programme qui résoudrait un problème correctement, mais mettrait trop longtemps à s'exécuter pourrait d'ailleurs voir son intérêt totalement anéanti[1].

On pourra en définitive adopter la définition suivante d'un programme : **suite ordonnée d'instructions permettant de résoudre efficacement et à tous les coups un problème clairement défini**.

Cette définition peut s'appliquer à de nombreuses situations de la vie quotidienne qui ne nécessitent pas l'utilisation d'un ordinateur. On peut, par exemple, penser au montage d'un meuble en kit ; cette opération s'apparente également à la réalisation d'un programme, et quiconque a tenté l'expérience sait que l'ordre de montage des éléments est primordial, si l'on veut aboutir au résultat recherché. Programmer consiste donc à écrire le scénario complet d'une tâche, en énumérant en détail des suites d'opérations, pour arriver à un résultat toujours identique. Exécuter un programme, c'est effectuer les unes après les autres les différentes étapes d'un scénario.

Algorithmique

Algorithmique est un mot qui fait un peu peur quand on ne sait pas vraiment de quoi il s'agit. L'algorithmique est une technique indissolublement liée à la programmation et c'est la raison pour laquelle nous allons la présenter.

Comme pour la programmation, commençons par examiner la définition qu'en donne le Robert :

« Science qui étudie l'application des algorithmes à l'informatique ».

Ce même dictionnaire définit l'algorithme en ces termes :

« Ensemble des règles opératoires propres à un calcul ou à un traitement informatique ».

1. Dans sa passionnante *Histoire des codes secrets*, Simon Singh raconte les exploits du célèbre et génial mathématicien anglais, Alan Turing, qui était chargé de créer un programme pour casser le code de la machine à chiffrer allemande, Enigma. Les premières versions de son programme fonctionnaient, mais elles étaient trop lentes pour trouver le code secret qui changeait tous les jours…

La définition est complétée par « calcul, enchaînement des actions nécessaires à l'accomplissement d'une tâche » et renvoie à la notion d'automate. Le Robert donne l'exemple de l'algorithme d'Euclide qui permet de trouver le plus grand commun diviseur (PGCD) de deux nombres[2]. Euclide ayant écrit ses traités à une période que l'on situe habituellement aux environs de 300 avant Jésus-Christ, on peut en conclure que les algorithmes existaient bien avant l'invention de l'informatique. Les algorithmes sont liés aux mathématiques et le nom algorithme vient d'ailleurs d'al-Khwârizmï[3] (parfois orthographié Al-Khawarizmi ou al-Khuwārizmī) qui était un savant perse ayant vécu à la fin du VIII^{ème} et dans la première moitié du IX^{ème} siècle. On doit notamment à ce grand mathématicien l'introduction des chiffres arabes en Occident.

Un algorithme est donc tout simplement une suite d'actions permettant d'accomplir une tâche. En d'autres termes, on peut également considérer qu'un algorithme est une méthode pour résoudre un problème en un nombre d'étapes fini. Il existe des algorithmes depuis l'Antiquité car très tôt on a eu besoin de calculer et de compter (le bétail, les impôts, etc.), mais l'invention de l'informatique dans la deuxième moitié du XX^{ème} siècle a remis au goût du jour la notion d'algorithme.

Encore une fois, la notion d'algorithme, même si elle est née dans le domaine des mathématiques et qu'elle est omniprésente en informatique, peut avoir des implications dans d'autres domaines qui n'utilisent pas forcément des machines. Il existe, par exemple, d'innombrables algorithmes dans le domaine des jeux. Les méthodes de résolution de jeux tels que le Rubik's cube, les tours de Hanoï, le taquin sont des algorithmes qui permettent de réussir à tous les coups ces casse-têtes.

Différences entre algorithme et programme

Maintenant que les notions d'algorithme et de programme sont plus précises dans votre esprit, vous vous demandez peut-être ce qui différencie finalement un algorithme d'un programme et si ces deux termes ne sont pas interchangeables.

Même s'il existe de grandes similitudes entre ces deux mots, ils ne désignent pas tout à fait la même chose. Un programme utilise des algorithmes et avant d'écrire un programme, il convient de déterminer les algorithmes qui permettront d'atteindre le résultat recherché. Un algorithme décrit de manière détaillée un traitement qui sera exécuté dans un programme et il est rare qu'un programme ne soit constitué que d'un seul algorithme. Un programme, même simple, peut contenir plusieurs algorithmes différents. Imaginons, par exemple, que vous souhaitiez réaliser le dictionnaire du texte d'un document Word que vous avez écrit, c'est-à-dire obtenir la liste de tous les mots que vous avez employés (moins certains mots qui n'ont pas de sens, comme les articles, les prépositions, les adverbes, etc.) et que cette liste doive être classée par ordre alphabétique. Il va falloir au minimum trois algorithmique : le premier pour délimiter les mots du texte et les isoler, le deuxième pour exclure les mots vides de sens et le troisième pour classer par ordre alphabétique les mots que vous aurez conservés.

2. Par exemple, le PGCD de 20 et 30 est 10. Il s'agit du plus grand nombre capable de diviser à la fois 20 et 30.

3. En latin médiéval, son nom a été latinisé en Algorithmus. Le terme algèbre vient également du nom de ce mathématicien qui a vécu à Bagdad.

Un programme ne se réduit pas seulement à ses algorithmes car il s'agit d'un ensemble complet comprenant ce que les informaticiens appellent dialogue homme-machine, et que l'on nomme aussi interface utilisateur. Ainsi la conception d'un formulaire ou bien l'affichage d'informations sur la page d'un écran ou sur une page imprimée font partie intégrante d'un programme, mais il n'y a pas vraiment d'algorithme sous-jacent, même s'il existe de nombreuses règles de conception pour élaborer des interfaces intuitives et conviviales.

Finalement, la plus grande différence entre un algorithme et un programme est le langage dans lequel il sera écrit. Les algorithmes sont toujours décrits dans un langage naturel (le français pour ce qui nous concerne) et si nous souhaitons écrire l'algorithme pour déterminer si un nombre est pair, on peut utiliser la formulation suivante :

Réaliser une division euclidienne[4] du nombre par deux. Si le reste est égal à 0, le nombre est pair, sinon il est impair.

Alors que si l'on veut coder cet algorithme dans un programme, il faudra utiliser un langage de programmation (sujet qui sera l'objet de la prochaine section). Un algorithme est donc indépendant de tout langage de programmation. Quand on veut programmer, on doit déterminer le but du programme et le résultat à atteindre ; il faut ensuite définir les algorithmes nécessaires aux traitements à mettre en œuvre, puis on doit traduire les algorithmes dans un langage de programmation.

Les langages de programmation

Un langage de programmation sert à traduire un algorithme en une suite d'instructions compréhensibles par un ordinateur qui est une machine ne sachant fondamentalement gérer que des suites de 0 et 1.

Comme l'ordinateur ne comprend pas (encore parfaitement) le langage naturel et qu'il est très difficile pour un être humain de programmer une machine en écrivant des suites de 0 et de 1, on a inventé des langages intermédiaires qui permettent d'écrire des programmes dans un langage plus simple que le code binaire. Il existe des centaines de langages de programmation, mais quelques langages sont très employés comme C, C++, Java, JavaScript, PHP, Python ou Visual Basic.

Avant d'étudier en détail la programmation VBA, nous allons vous montrer un exemple de programme écrit dans trois langages de programmation différents. Même si vous ne comprenez pas la syntaxe de ces programmes et que vous êtes pour l'instant incapable de les reproduire ou de les imiter, vous constaterez néanmoins leurs différences et leurs similitudes. À des fins pédagogiques, nous avons choisi un exemple de programme simplissime qui consiste à demander à l'utilisateur d'indiquer l'âge d'une personne, le programme se chargeant de déterminer si la personne est mineure ou majeure.

Commençons par suivre les conseils que nous avons prodigués et déterminons l'algorithme du programme que l'on peut formuler de la manière suivante :

- Poser une question à l'utilisateur
- Récupérer l'information saisie par l'utilisateur

4. Pour mémoire, une division euclidienne, également appelée division entière, ne fonctionne qu'avec des entiers. Si je divise 35 par 4, le résultat est 8 et le reste est égal à 3.

- Si le nombre saisi est strictement inférieur à 18, alors la personne est mineure, sinon elle est majeure.

Comme le programme est très simple, l'algorithme est très court et se réduit à une comparaison d'un nombre par rapport à une valeur de seuil (l'âge de la majorité).

Voici la traduction de cet algorithme en JavaScript qui est un langage de programmation dédié aux pages Web.

```
var age;
age = parseFloat(window.prompt('Quel est votre âge ?'));
if (age < 18) {
  window.alert('Vous êtes une personne mineure');
} else {
  window.alert('Vous êtes une personne majeure');
}
```

La version suivante est écrite en Python, un langage polyvalent qui est étudié au lycée :

```
age = None
age = float(input('Quel est votre âge ?'))
if age < 18:
  print('Vous êtes une personne mineure')
else:
  print('Vous êtes une personne majeure')
```

La troisième version emploie le langage VBA (Visual Basic pour Applications) qui est l'objet de cet ouvrage.

```
Dim age As Byte
age = InputBox("Quel âge avez-vous ?")
If age < 18 Then
MsgBox ("Vous êtes une personne mineure")
Else
MsgBox ("Vous êtes une personne majeure")
End If
```

Même si vous ne comprenez pas parfaitement ces programmes, vous pouvez constater qu'ils ont une structure similaire : la première ligne évoque le mot âge (qui est écrit age dans le programme), puis on pose la question à l'utilisateur pour qu'il puisse saisir sa réponse, et enfin on compare la valeur de l'âge à 18. Il existe des différences de vocabulaire entre les trois langages de programmation, notamment quand on demande à l'utilisateur de saisir l'âge de la personne (`window.prompt`, `input` et `inputbox`), mais on retrouve dans la comparaison des similitudes, les trois langages employant les mots *if* (si en anglais) et *else* (sinon en anglais). Il existe également des différences dans la manière de délimiter les phrases et certains langages utilisent des accolades, ou bien encore les caractères point-virgule ou deux-points.

Terminologie de la programmation

Afin de bien commencer l'apprentissage de la programmation, il est nécessaire de fixer le sens de certains termes qui reviennent très souvent quand on programme et qui peuvent avoir un sens différent dans la langue courante.

Il faut tout d'abord noter une grande analogie entre les langages naturels (le français, l'anglais, l'allemand, etc.) et les langages de programmation, si bien que des termes propres à l'apprentissage des langues se retrouvent en informatique. Ainsi, on parle de la syntaxe (la grammaire) des langages de programmation et des erreurs de syntaxe dans un programme. Il existe cependant une différence majeure dans la syntaxe des langages de programmation par rapport à celle des langues naturelles : elle est stricte et ne tolère aucune approximation. À la différence d'un être humain qui comprendra assez facilement un énoncé incorrect (« vous voyez ça que je veux dire »), un programme informatique ne fonctionnera pas si vous employez une virgule à la place d'un point-virgule. Cette rigueur est très souvent déroutante et frustrante pour celui qui débute l'apprentissage de la programmation, mais elle est également formatrice.

Comme avec une langue naturelle, apprendre un langage de programmation, c'est finalement apprendre la grammaire (syntaxe) et le vocabulaire de cette langue un peu particulière qui permet d'exécuter des programmes. Il y a plusieurs catégories de mots dans un langage de programmation et on retrouvera ce qui peut s'apparenter à des verbes, des adjectifs, des conjonctions de coordination, etc. Chaque langage de programmation possède sa propre syntaxe et son propre vocabulaire, même s'il existe de grandes similitudes entre certains langages de programmation.

L'ensemble des lignes d'un programme forme ce que l'on appelle le **code** du programme, que l'on nomme aussi **code source**, voire simplement **source**. On parle de code car on considère que la traduction de l'algorithme en langage de programmation constitue une forme d'encodage, si bien que coder et programmer sont des synonymes. Par voie de conséquence, un **codeur** est un programmeur, c'est-à-dire une personne qui écrit des programmes. On notera que **développeur** est un autre synonyme de codeur. **Coder** (ou encoder) consiste donc à traduire les actions à exécuter dans un langage de programmation.

> Bien évidemment, on emploie ici le masculin générique et il va sans dire que la programmation s'adresse aussi bien aux hommes qu'aux femmes. Il existe par conséquent d'excellentes codeuses et il faut ici rappeler que le premier programmeur de l'histoire a été une femme, Ada Lovelace :
> https://www.franceculture.fr/numerique/ada-lovelace-la-premiere-codeuse-de-lhistoire

Ces actions, définies par des algorithmes, sont souvent rédigées dans un langage que l'on appelle **pseudo-code**, qui est un intermédiaire entre la description en langage naturel et le langage de programmation. Il s'agit d'une forme de langage naturel où l'on s'autorise une écriture abrégée. Si l'on reprend l'exemple du programme dont nous avons écrit trois versions, voici ce que donnerait le pseudo-code :

```
saisie utilisateur
si saisie < 18
afficher mineur
sinon afficher majeur
```

Le code d'un programme est constitué de phrases élémentaires que l'on appelle lignes d'instruction, chaque ligne d'instruction exécutant une action élémentaire.

Quand un programme ne fonctionne pas correctement, on dit familièrement qu'il « se plante ». Cela signifie souvent qu'il s'arrête brutalement, les raisons de ce plantage étant en général dues à une erreur de programmation que l'on appelle **bug** en informatique. Ce mot anglais, qui signifie notamment insecte, vient du fait que les premiers dispositifs électroniques pouvaient être perturbés par des insectes attirés par la lumière qu'ils émettaient. Aujourd'hui, un bug est devenu une erreur dans un programme et on continue à utiliser le terme anglais, même si l'équivalent français « bogue » a été proposé.

VBA ET LES LANGAGES DE PROGRAMMATION

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. 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 était présente dans Office 2000, Office XP, Office 2003 et Office 2007. Office 2010 a accueilli la version 7 de VBA et c'est la version 7.1 qui tourne dans Office 2013 et Office 2016, Microsoft ayant annoncé son intention de ne plus faire évoluer ce langage qui est donc arrivé à maturité.

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 logiciels, comme Autocad, peuvent être programmés en 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