FORMATION À VBA Office 2013

Dominique Maniez

DUNOD

Toutes les marques citées dans cet ouvrage sont
des marques déposées par leurs propriétaires respectifs.

ISBN 978-2-10-058941-8

Table des matières

Partie 2 – Le langage VBA

Partie 4 – Programmation VBA avancée

Avant-propos

Il y a une dizaine d'années, quand j'ai écrit mon premier livre sur VBA (Visual Basic pour Applications), mon objectif était simple : *faire découvrir aux lecteurs francophones* la simplicité et la puissance de ce langage de programmation qui était à la disposition de tout utilisateur de Word et d'Excel. Sans vouloir me vanter, je crois que mon but a été atteint et c'est avec un réel plaisir que je mets à jour cet ouvrage chaque fois que sort une nouvelle version d'Office.

Si les versions d'Office se suivent et ne se ressemblent pas toutes (l'adoption du ruban a parfois été vécue douloureusement…), VBA est toujours bien présent dans la version estampillée 2013 et je souhaite démontrer à tous les utilisateurs d'Office qu'ils se privent inutilement de la richesse fonctionnelle de leur traitement de texte ou de leur tableur en ignorant la programmation. En vous apprenant à programmer Word, Excel, Access, Outlook et PowerPoint, je veux tout d'abord vous montrer que cette activité n'est pas réservée aux professionnels de l'informatique et vous faire gagner du temps dans l'exécution des tâches répétitives et fastidieuses.

À QUI S'ADRESSE CE LIVRE ?

Cet ouvrage est un livre d'initiation et il ne nécessite donc aucune connaissance préalable en programmation ; il vise par conséquent un public de débutants. Il s'adresse en priorité aux utilisateurs de la suite Office qui souhaitent apprendre la programmation afin d'améliorer leur productivité. Les personnes utilisant Office et possédant déjà une expérience de programmeur peuvent également profiter de ce livre, mais négliger la lecture des chapitres consacrés aux rudiments de la programmation. Cet ouvrage n'est pas un ouvrage de référence et il ne prétend donc pas à l'exhaustivité ; de nombreuses informations sont sciemment passées sous silence afin de clarifier le propos et de ne pas semer

la confusion dans l'esprit du lecteur par un apport trop important de connaissances nouvelles.

La démarche pédagogique mise en œuvre dans ce livre est similaire à la méthode de programmation par raffinements successifs ; cette méthode reprend en fait un principe cartésien qui stipule qu'il faut commencer « par les objets les plus simples et les plus aisés à connaître, pour monter peu à peu, comme par degrés, jusqu'à la connaissance des plus composés ». La dernière partie de cet ouvrage propose à ceux qui le souhaitent, des pistes pour qu'ils puissent approfondir les sujets abordés dans ces pages ou bien explorer d'autres horizons plus complexes de la programmation sous Office.

POURQUOI APPRENDRE À PROGRAMMER OFFICE ?

La première raison est productiviste. La programmation, même à un niveau peu élevé, va vous permettre de gagner un temps précieux, surtout si vous accomplissez des tâches répétitives. En effet, l'automatisation des tâches va augmenter votre productivité, parfois dans des proportions que vous n'imaginez même pas. Outre le gain de temps, vous allez également vous affranchir des tâches pénibles et pouvoir ainsi vous consacrer à des tâches plus nobles. En définitive, vous constaterez que l'amélioration est non seulement quantitative, mais également qualitative.

La deuxième raison est qu'en programmant vous allez pouvoir bénéficier d'un logiciel sur mesure car vous allez créer tout ce qui vous manque. Les possibilités de paramétrage d'Office sont déjà importantes, mais en programmant, vous allez contrôler exactement les traitements de votre système d'information. Apprendre à programmer ouvre des horizons quasiment infinis et il est impossible d'inventorier toutes les applications pratiques. En maîtrisant les rudiments de la programmation, vous allez déjà pouvoir inventer des commandes et des fonctions qui n'existent pas dans le logiciel (par exemple des fonctions d'Excel qui vous manquent). Vous allez aussi pouvoir contrôler la validité des informations qui sont saisies dans Excel ou Access. Dans tous ces logiciels, il est extrêmement facile de saisir des données mais dès que l'on veut exercer un contrôle minimal sur les informations qui sont saisies, il faut avoir recours à la programmation. Et si l'on réfléchit bien, on s'aperçoit qu'il est inutile de traiter des données par de savants calculs si l'on n'a

pas pris la précaution de s'assurer de la validité de ces informations. De la même manière, si vous développez des modèles qui doivent être utilisés par d'autres, la programmation vous aidera à définir des écrans d'aide spécifiques ou bien des formulaires de saisie personnalisés qui faciliteront la tâche de ceux qui doivent entrer les informations.

La dernière raison est d'ordre intellectuel. 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. En même temps, on acquiert certains types de raisonnements logiques qui peuvent servir dans d'autres domaines que celui de la programmation.

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 sous Office ;
- vous pourrez vous lancer dans l'apprentissage d'un langage de programmation plus puissant.

Importance des exemples de code

Il est impossible de concevoir un ouvrage traitant de la programmation Office sans de nombreux exemples de code car, si l'on apprend à programmer en programmant, on étudie également la programmation en examinant le code de programmes écrits par d'autres. Imprimer le code de tous les exemples au sein de cet ouvrage ne serait guère raisonnable car cela prendrait une place considérable ; il est d'autre part prouvé que la recopie d'un listing imprimé engendre de nombreuses erreurs de retranscriptions. C'est pour cette raison que ne sont imprimés dans ce livre que de courts exemples ou bien des extraits de programmes plus longs. Il est absolument nécessaire que vous vous procuriez la totalité des exemples de code de cet ouvrage qui sont disponibles sur *www.dunod.com* à la page dédiée à l'ouvrage, ou bien sur mon site personnel *www.cosi.fr* dans la rubrique Code des ouvrages.

UN OUVRAGE VRAIMENT CONÇU POUR LES DÉBUTANTS

C'est peut-être parce que je n'arrivais pas à trouver les livres que j'avais envie de lire que je me suis mis à en écrire. Cela ne veut pas dire que mes livres sont meilleurs que les autres, mais tout simplement qu'ils correspondent mieux à mon mode d'apprentissage.

Quand j'ai commencé à apprendre à programmer au début des années 1980, j'ai dévoré des dizaines de livres sur le sujet. Après toutes ces années passées à lire cette littérature technique sur la programmation, je suis arrivé à la conclusion qu'il n'existait pas véritablement d'ouvrage conçu pour les débutants qui n'y connaissent rien du tout, les livres de programmation étant avant tout conçus pour les informaticiens. Comme mon credo est que tout le monde peut programmer et que la programmation ne doit surtout pas être réservée aux informaticiens, il existe un véritable problème pour les gens qui ne sont pas informaticiens et qui souhaitent néanmoins s'initier à la programmation. Ce livre s'adresse à ces personnes qui veulent découvrir les joies (et les peines) de la programmation avec Office. Cet objectif implique que la pédagogie mise en œuvre dans cet ouvrage prenne véritablement en compte le manque d'expérience du lecteur. Je ne prendrai qu'un seul exemple qui illustre bien cette différence de traitement pédagogique ; dans les livres de programmation, il est nécessaire d'apprendre la syntaxe (c'est-à-dire la grammaire) du langage de programmation étudié. En général, tous les livres commencent par décrire la syntaxe formelle, puis prennent des exemples. Nous sommes persuadés que cette méthode ne fonctionne pas avec des débutants qui ne sont pas habitués au formalisme de la description de la syntaxe du langage. Nous pensons au contraire qu'il faut commencer par les exemples et éventuellement passer au formalisme, après avoir étudié de nombreux exemples.

POURQUOI APPRENDRE LA PROGRAMMATION DE CINQ LOGICIELS EN MÊME TEMPS ?

Dans les premières versions d'Office, chaque logiciel de la suite avait son propre langage et les langages de programmation étaient donc incompatibles entre eux ; ainsi, par exemple, Word Basic n'était pas compatible avec Access Basic. Avec l'avènement d'Office 2000, Microsoft a réalisé un effort considérable d'harmonisation et VBA est maintenant l'uni-

que langage de programmation de la suite. Cela signifie que quand j'apprends à programmer Word, je sais programmer à la fois Excel, Access, Outlook et PowerPoint. L'unicité de ce langage est un progrès énorme et c'est pour cette raison qu'il serait dommage de se limiter à l'apprentissage de la programmation d'un seul logiciel quand il est si facile de passer d'un logiciel à l'autre.

Le fait qu'Office propose un même langage pour toutes ses applications est réellement un avantage déterminant et nous pensons qu'il va inciter plus d'un utilisateur à se lancer dans l'aventure de l'apprentissage de la programmation VBA.

COMMENT APPRENDRE À PROGRAMMER OFFICE ?

Au risque de rappeler une évidence, pour apprendre à programmer Office, il faut déjà apprendre Office. Cette vérité première mérite d'être répétée tant on a vu d'utilisateurs se lancer dans l'apprentissage de la programmation sans maîtriser les fonctionnalités élémentaires de Word (par exemple, les styles, les modèles ou bien encore les tableaux), d'Excel (écriture d'une formule, adresse relative ou absolue, etc.) ou d'Access (création de tables, de requêtes ou de formulaires). Si vous pensez que vos connaissances d'Office sont imparfaites, vous devrez donc au préalable les approfondir.

Une fois que ces connaissances sont acquises, il faut apprendre le langage de programmation VBA et le modèle d'objets des applications Office. Nous emploierons ici souvent l'analogie avec l'apprentissage des langues vivantes et l'ambition de ce livre est donc de vous enseigner la syntaxe (le langage VBA) et le vocabulaire (le modèle d'objets) de chacun des logiciels de la suite afin que vous puissiez écrire vous-même rapidement des programmes.

Il existe cependant une difficulté importante quand on veut apprendre une langue étrangère : par où commencer ? La tâche est immense et la logique voudrait qu'avant de s'exprimer on commence par maîtriser la grammaire et le vocabulaire. Mais cette approche pédagogique est bien peu efficace et chacun d'entre nous se rend compte que l'on n'apprend bien une langue qu'en la pratiquant, la théorie ne pouvant venir que dans un deuxième temps.

Nous allons donc apprendre à programmer en programmant et nous étudierons la théorie seulement quand nous en aurons réellement besoin.

QUELLE VERSION D'OFFICE FAUT-IL UTILISER AVEC CE LIVRE ?

Tous les exemples de cet ouvrage ont été conçus et testés avec la version 2013 d'Office sous Windows 8. Il est possible qu'ils fonctionnent également avec les versions précédentes, mais nous ne pouvons le garantir.

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, n'importe qui maîtrisant 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 cette définition plus complète : « *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.

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'instructions est une opération assez banale dans la vie quotidienne et quand 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 g 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 g de saumon fumé et mixer le tout. Rajouter 400 g 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 5 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 cuisine 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 dans un langage de programmation plusieurs catégories de mots (verbe, adjectif, conjonction de coordination, etc.) 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 (on a additionné puis multiplié) ou 14 (on a multiplié puis additionné). 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 sortie. 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 a accueilli la version 7 de VBA et c'est la version 7.1 qui tourne dans Office 2013.

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.

Compilation

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 (Visual Basic pour Applications) est le langage de programmation d'Office. Toutes les applications Office (sauf OneNote) intègrent un interpréteur VBA qui permet d'exécuter des programmes écrits dans ce langage.

2

Enregistrer une macro

La documentation de Word définit une macro comme une série de commandes et d'instructions regroupées au sein d'une même commande afin d'exécuter automatiquement une tâche. Pour Excel, une macro est une série de commandes et de fonctions stockées dans un module Visual Basic, qui peut être exécutée chaque fois qu'on doit accomplir cette tâche. Nous allons voir dans ce chapitre qu'il est très simple d'écrire ses premières macros en utilisant l'enregistreur de macro.

Il y a une vingtaine d'années, Microsoft inventa pour ses logiciels Word et Multiplan (l'ancêtre d'Excel) le concept de macro-commande. Il s'agissait de la possibilité de mémoriser les touches frappées au clavier, les options sélectionnées et les commandes exécutées afin de les réutiliser plus tard. L'utilisateur avait donc la possibilité d'enregistrer une suite de commandes du logiciel pour automatiser les actions les plus répétitives. Mais l'écriture de macro-commandes était assez complexe et le mini langage de programmation qui accompagnait Word et Multiplan était assez pauvre.

Aujourd'hui, avec Office, les choses ont considérablement évolué et de la même manière que l'on ne parle plus de micro-informatique, mais de micro, les macro-commandes sont devenues les macros. L'utilisateur de la suite Office dispose à présent d'un langage de programmation puissant et complet, VBA, qui est doté d'un environnement digne des langages utilisés par les informaticiens professionnels.

L'ENREGISTREUR DE MACRO

Word et Excel disposent d'un enregistreur de macro qui, à la manière d'un magnétophone, peut enregistrer vos actions dans le logiciel et rejouer à volonté ce que vous avez exécuté.

Remarque

Les autres logiciels de la suite Office (Access, PowerPoint, Outlook, etc.) ne possèdent pas d'enregistreur de macro et le code VBA ne peut donc pas être généré automatiquement. En revanche, il existe dans Access un type d'objet nommé macro qui permet de stocker séquentiellement une série d'actions à accomplir ; cependant, les macros de ce type n'utilisent pas le langage VBA.

Quand devez-vous enregistrer une macro ?

Chaque fois que vous réalisez une tâche répétitive dans Word ou Excel, vous devez vous poser la question de l'intérêt d'une macro. Il n'est nul besoin que la tâche à accomplir soit excessivement longue ; il suffit simplement que vous l'accomplissiez souvent. Si, par exemple, vous devez tous les jours imprimer la dixième page d'un document, vous pouvez enregistrer une macro qui automatisera cette tâche. Même si le temps gagné est en l'occurrence minime (une dizaine de secondes), vous devez systématiser cette démarche qui vous permettra au final d'économiser un temps appréciable.

En écrivant des macros, vous allez travailler plus intelligemment et puis vous gagnerez en efficacité car, quand une macro fonctionne bien, elle fonctionne bien tout le temps.

A contrario, il ne sert à rien d'enregistrer une macro pour une tâche que vous n'accomplissez qu'une seule fois ou de manière très épisodique. Même si le fait d'enregistrer une macro n'est pas complexe et ne prend que quelques secondes en plus, il est inutile de le faire si vous n'avez pas l'occasion d'exploiter la macro enregistrée.

Conseil

Si vous effectuez souvent la même mise en forme sous Word (par exemple, une modification de la police, un changement de la taille de la police et une mise en gras), il est préférable de créer un style plutôt que d'enregistrer une macro. Si jamais le style défini ne vous convient plus, une seule modification du style suffira à changer automatiquement toutes les occurrences de ce style dans l'ensemble du document. En revanche, avec une macro, il faudrait non seulement modifier la macro, mais l'exécuter à nouveau sur tout le document pour chaque occurrence du style. En pareil cas, une commande de recherche et de remplacement serait d'ailleurs plus efficace.

Enregistrement de votre première macro

Imaginez, par exemple, que vous deviez souvent remettre en forme des documents Word dans lesquels l'utilisateur n'a pas cru bon de saisir un espace insécable avant le caractère deux-points. Pour ce faire, une simple commande de recherche et de remplacement fait l'affaire et cette opération n'est pas très longue, mais si elle doit se répéter souvent, elle deviendra vite fastidieuse. Nous allons voir comment nous pouvons facilement l'automatiser grâce à une macro.

Remarque

Dans un document de traitement de texte, vous devez saisir un espace insécable avant les caractères deux-points, point-virgule, point d'interrogation, point d'exclamation, ainsi qu'entre les mots qui forment un groupe qui ne doit pas être coupé, comme dans « Louis XVI » ou bien encore les dates et les nombres. Pour visualiser les espaces et les espaces insécables d'un document Word, utilisez la commande Afficher tout qui se trouve dans le groupe Paragraphe de l'onglet Accueil (symbole de la pompe à essence, le raccourci clavier étant CTRL + Majuscule + 8).

Pour faire l'exercice, lancez Word et cliquez dans l'**onglet Développeur** sur la commande **Enregistrer une macro**.

Astuce

Si l'onglet Développeur ne figure pas sur le ruban de Word, utilisez la commande Fichier→Options→Personnaliser le ruban et cochez la case Développeur dans la liste Onglets principaux.