# Excel et les fichiers de données

## Travailler avec des fichiers texte

Bien qu'il puisse largement le faire, Excel n'a pas été conçu pour stocker de grandes masses de données. Pour cela, les logiciels les plus adaptés sont les gestionnaires de bases de données comme, par exemple, Access.

Mais les gestionnaires de base de données les plus performants ne sont généralement pas gratuits. Heureusement, Excel propose une alternative en permettant de travailler sur des fichiers, notamment des fichiers texte.

Les fichiers texte sont des fichiers contenant uniquement des caractères imprimables et des retours à la ligne. Leur format est reconnu universellement si bien qu'ils présentent l'immense avantage de pouvoir être utilisés par de très nombreux logiciels.

### *L'instruction Open*

Il est possible de créer des fichiers texte avec Visual Basic grâce à l'instruction Open. Deux écritures sont possibles :

Open *fichier* for Output as #n

Open *fichier* for Append as #n

Dans les deux cas, *fichier* désigne un nom de fichier avec son adresse, par exemple *C:\Dossier\essai.txt*, et n est un entier compris entre 1 et 511 inclus qui servira à désigner le fichier.

Si un fichier a été ouvert en mode Output ou Append, il est possible de lui ajouter des données.

Avec l'option Output, si le fichier existe déjà, son contenu est effacé ; avec l'option Append, les nouveaux enregistrements viennent s'ajouter aux enregistrements existants.

Après avoir utilisé un fichier texte, il faut impérativement le fermer avec l'instruction Close #n, où n est l'entier utilisé pour son ouverture.

Pour pouvoir lire les données d'un fichier texte, il faut d'abord l'ouvrir avec l'instruction Open en mode Input :

Open *fichier* for Input as #n

## L'instruction Print

La méthode la plus simple pour écrire dans un fichier texte est d'utiliser l'instruction Print. Elle permet d'écrire une ligne de texte dans un fichier ouvert par l'instruction Open en mode Output ou Append. Elle s'écrit ainsi :

Print #n, Texte

Où n désigne l'entier utilisé pour ouvrir le fichier et Texte la ligne de texte que l'on désire entrer dans le fichier.

Un exemple de programme est le suivant :

```
Sub Textes()
Open "C:\Essais\FichierTexte.txt" For Output As #1
Print #1, "Bonjour."
Print #1, "Nous allons travailler sur les fichiers texte."
Close #1
End Sub
```

Pour vérifier le résultat, le mieux est d'ouvrir dans le dossier C:\Essais le fichier FichierTexte.txt avec le bloc-notes de Windows. Nous obtenons :

```
Bonjour.
Nous allons travailler sur les fichiers texte.
```

## L'instruction Line Input

Pour lire les données du fichier FichierTexte.txt avec Visual Basic, nous devons d'abord l'ouvrir en mode Input puis nous pouvons utiliser l'instruction Line Input. Cette instruction permet de lire une ligne du fichier, sa syntaxe est la suivante :

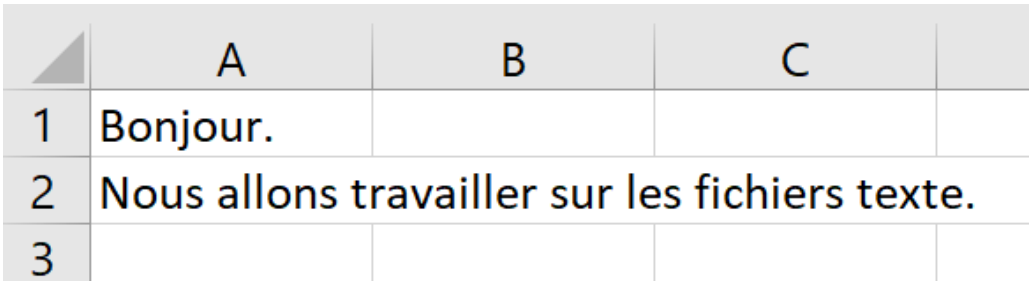
Line Input #n, VarTexte

Où n désigne l'entier utilisé pour ouvrir le fichier et VarTexte la variable qui recueillera la ligne de texte du fichier. Un exemple de programme est le suivant :

```
Sub Lire()
Dim enr As String
Set f = ThisWorkbook.Sheets(1)
Open "C:\Essais\FichierTexte.txt" For Input As #1
Line Input #1, enr
f.Range("A1") = enr
Line Input #1, enr
```

```
f.Range("A2") = enr
Close #1
End Sub
```

Nous obtenons :



| | A | B | C |
|---|---|---|---|
| 1 | Bonjour. | | |
| 2 | Nous allons travailler sur les fichiers texte. | | |
| 3 | | | |

Dans ce programme, la première instruction Line Input lit la première ligne du fichier et la place dans la variable *enr*. Le curseur se déplace à la deuxième ligne qui est donc lue par la deuxième instruction Line Input.

Ce programme nous a permis de lire les deux premières lignes du fichier mais, en général, nous voulons lire tout le fichier et nous ne savons pas combien il possède de lignes. Nous allons donc le lire avec une boucle comme dans le programme suivant :

```
Sub Lire()
Dim enr As String
Set f = ThisWorkbook.Sheets(1)
Open "C:\Essais\FichierTexte.txt" For Input As #1
i = 1
Do While Not EOF(1)
    Line Input #1, enr
    f.Cells(i, 1) = enr
    i = i + 1
Loop
Close #1
End Sub
```

Ce programme utilise la fonction EOF() qui prend la valeur TRUE lorsqu'on arrive à la dernière ligne.

## Enregistrements de longueur constante

Nous avons vu comment stocker des lignes de texte dans un fichier mais nous pouvons aller plus loin et stocker des données structurées. Supposons, par exemple, que nous voulions enregistrer les ventes de différents produits réalisées par deux vendeurs comme dans le tableau suivant :

| | A | B | C |
|---|---|---|---|
| 1 | Robes | Martin | 1234,12 |
| 2 | Manteaux | Martin | 675,32 |
| 3 | Pantalons | Martin | 3455,64 |
| 4 | Chemises | Martin | 2145,12 |
| 5 | Robes | Dupond | 3421,12 |
| 6 | Manteaux | Dupond | 2541,34 |
| 7 | Pantalons | Dupond | 2365,15 |
| 8 | Chemises | Dupond | 3215,37 |
| 9 | | | |

Nous pouvons créer des enregistrements de longueur constante comme dans le programme suivant :

```
Sub EcrireConst()
Dim produit As String * 20
Dim vendeur As String * 20
Dim ventes As String * 10
Dim enr As String * 50
Set f = ThisWorkbook.Sheets(1)
Open "C:\Essais\FichierTexte.txt" For Output As #1
For i = 1 To 8
    produit = f.Cells(i, 1)
    vendeur = f.Cells(i, 2)
    ventes = f.Cells(i, 3)
    enr = produit & vendeur & ventes
    Print #1, enr
Next i
Close #1
End Sub
```

Comme nous avons défini les trois variables comme étant des textes de longueur fixée, la longueur de l'enregistrement *enr* est égale à 20 + 20 + 10 = 50.

Nous pouvons vérifier avec le bloc-notes que le fichier a bien été saisi mais nous pouvons aussi le lire avec un programme Visual Basic.

## *Lire le fichier texte avec Visual Basic*

Comme tous les enregistrements ont la même longueur, nous pouvons pour chaque ligne lire séparément le produit, le vendeur et la vente, par exemple avec le programme suivant :

```
Sub LireConst()
Dim produit As String * 20
Dim vendeur As String * 20
Dim ventes As String * 10
Dim enr As String * 50
Set f = ThisWorkbook.Sheets(1)
Open "C:\Essais\FichierTexte.txt" For Input As #1
i = 20
Do While Not EOF(1)
    Line Input #1, enr
    produit = Mid(enr, 1, 20)
    vendeur = Mid(enr, 21, 20)
    ventes = Mid(enr, 41, 10)
    ventes = Replace(ventes, ",", ".")
    va = Val(ventes)
    f.Cells(i, 1) = produit
    f.Cells(i, 2) = vendeur
    f.Cells(i, 3) = va
    i = i + 1
Loop
Close #1
End Sub
```

Dans ce programme, nous avons utilisé la fonction Mid(Texte, Debut, Longueur) qui extrait de *Texte* une chaîne de caractères à partir de la position *Debut* et sur une longueur égale à *Longueur*.

Comme nous voulons pouvoir faire des calculs sur les ventes nous avons converti le texte correspondant aux ventes en nombre. Pour cela, nous avons commencé par remplacer la virgule des nombres par un point puisque Visual Basic utilise le point comme séparateur décimal. Nous avons ensuite utilisé la fonction *Val* pour convertir le texte en nombre.

Nous pouvons maintenant calculer la commission des deux vendeurs en supposant qu'elle est égale à 20% grâce au programme suivant :

```
Sub Commissions()
Dim produit As String * 20
Dim vendeur As String * 20
Dim ventes As String * 10
Dim enr As String * 50
Dim comM, comD As Double
Set f = ThisWorkbook.Sheets(1)
Open "C:\Essais\FichierTexte.txt" For Input As #1
comM = 0
comD = 0
Do While Not EOF(1)
    Line Input #1, enr
    vendeur = Mid(enr, 21, 20)
    ventes = Mid(enr, 41, 10)
    ventes = Replace(ventes, ",", ".")
    va = Val(ventes) * 0.2
    If Trim(vendeur) = "Martin" Then comM = comM + va
    If Trim(vendeur) = "Dupond" Then comD = comD + va
Loop
f.Cells(10, 1) = "Martin"
f.Cells(10, 2) = comM
f.Cells(11, 1) = "Dupond"
f.Cells(11, 2) = comD Close #1
End Sub
```

On obtient :

| | | |
|---|---|---|
| 10 | Martin | 1502,04 |
| 11 | Dupond | 2308,596 |

## *Lire un fichier texte avec un tableau croisé dynamique*

Le tableau croisé dynamique est outil d'Excel très performant et il peut s'avérer extrêmement utile pour lire un fichier texte.

Lisons-donc avec un tableau croisé dynamique le fichier *FichierTexte.txt* que nous avons créé. Pour cela, nous devons aller dans le menu *Données* d'Excel puis sélectionner *A partir d'un fichier texte/CSV*. Il faut alors sélectionner *FichierTexte.txt* puis cliquer sur *Importer*.

Il apparaît un encadré présentant notre fichier et nous proposant un délimiteur. Nous allons choisir *Largeur fixe* et Excel nous propose une

liste de chiffres commençant par zéro. Ces chiffres correspondent à la position de début de chaque colonne, nous allons donc entrer 0, 20, 40.

Dans la liste déroulante *Détection du type de données*, il est préférable de sélectionner *Selon le jeu de données complet* si toutes les valeurs n'ont pas le même nombre de décimales car, par exemple, si les 300 premières valeurs sont des entiers, Excel va comprendre que toutes les valeurs sont des entiers et il va convertir en nombre entier tous les nombres décimaux qu'il va trouver ensuite.

Dans le coin inférieur droit du cadre se trouve la liste déroulante *Charger*. Nous allons cliquer sur *Charger dans* puis sur *Rapport de tableau croisé dynamique*.

Nous pouvons alors personnaliser notre tableau croisé dynamique. Par exemple, nous pouvons mettre en ligne *Column1* qui correspond aux produits, en colonne *Column2* qui correspond aux vendeurs et en valeurs *Column3* qui correspond aux ventes. On obtient le tableau suivant :

| Somme de Column3 | Étiquettes de colonnes | | |
|---|---|---|---|
| Étiquettes de lignes | Dupond | Martin | Total général |
| Chemises | 3215,37 | 2145,12 | 5360,49 |
| Manteaux | 2541,34 | 675,32 | 3216,66 |
| Pantalons | 2365,15 | 3455,64 | 5820,79 |
| Robes | 3421,12 | 1234,12 | 4655,24 |
| **Total général** | **11542,98** | **7510,2** | **19053,18** |

Nous pouvons retrouver les commissions payées aux vendeurs en mettant en colonnes les vendeurs puis les produits et en introduisant un nouveau champ calculé que nous pouvons appeler *Commissions*. Pour cela, nous cliquons sur le tableau croisé dynamique et dans le menu *Analyse* nous sélectionnons *Calculs* puis *Champs, éléments et jeux*. Nous choisissons *Champ calculé* et nous introduisons *Commissions* comme nom et *Column3*20%* dans la formule.

Nous obtenons :

| Étiquettes de lignes | Somme de Column3 | Somme de Commissions |
|---|---|---|
| **Dupond** | **11542,98** | **2308,596** |
| Chemises | 3215,37 | 643,074 |
| Manteaux | 2541,34 | 508,268 |
| Pantalons | 2365,15 | 473,03 |
| Robes | 3421,12 | 684,224 |
| **Martin** | **7510,2** | **1502,04** |
| Chemises | 2145,12 | 429,024 |
| Manteaux | 675,32 | 135,064 |
| Pantalons | 3455,64 | 691,128 |
| Robes | 1234,12 | 246,824 |
| **Total général** | **19053,18** | **3810,636** |

Nous pouvons vérifier que nous obtenons les mêmes résultats qu'avec le programme Visual basic. Nous pouvons également personnaliser notre tableau, par exemple de la manière suivante :

| Vendeur / Produit | Ventes | Commissions |
|---|---|---|
| **Dupond** | **11 542,98** | **2 308,60** |
| Chemises | 3 215,37 | 643,07 |
| Manteaux | 2 541,34 | 508,27 |
| Pantalons | 2 365,15 | 473,03 |
| Robes | 3 421,12 | 684,22 |
| **Martin** | **7 510,20** | **1 502,04** |
| Chemises | 2 145,12 | 429,02 |
| Manteaux | 675,32 | 135,06 |
| Pantalons | 3 455,64 | 691,13 |
| Robes | 1 234,12 | 246,82 |
| **Total général** | **19 053,18** | **3 810,64** |

# Travailler avec des fichiers CSV

Les fichiers CSV sont des fichiers texte contenant des données séparées par une virgule ou point-virgule. La virgule correspond au format américain puisque le séparateur décimal est le point. En France, le séparateur est généralement le point-virgule car la virgule est utilisée comme séparateur décimal.

L'acronyme CSV provient du format américain et signifie *Comma Separator Values*. Les fichiers CSV permettent de stocker des données qui pourront être utilisées aussi bien par Excel que par n'importe quel gestionnaire de bases de données. Par rapport aux fichiers à enregistrements de longueur constante, ils présentent généralement l'avantage d'être plus compacts.

## *Créer des fichiers CSV*

Pour créer des fichiers CSV avec Visual Basic, nous pouvons procéder comme nous l'avons fait précédemment pour créer un fichier texte, il suffit simplement d'introduire le séparateur entre les données. Par exemple, si nous reprenons notre tableau des ventes, nous pouvons créer un fichier CSV utilisant le point-virgule comme séparateur avec le programme suivant :

```
Sub EcrireCSV()
Dim produit As String
Dim vendeur As String
Dim ventes As Double
Dim enr As String
Set f = ThisWorkbook.Sheets(1)
Open "C:\Essais\FichierCSV.csv" For Output As #1
For i = 1 To 8
    produit = f.Cells(i, 1)
    vendeur = f.Cells(i, 2)
    ventes = f.Cells(i, 3)
    enr = produit & ";" & vendeur & ";" & ventes
    Print #1, enr
Next i
Close #1
End Sub
```

## *Lire des fichiers CSV*

Nous pouvons vérifier avec le bloc-notes que nous avons obtenu le bon résultat. Nous pouvons également lire le fichier grâce à un programme Visual Basic comme celui-ci qui recopie les données dans la feuille 3 du classeur.

```
Sub LireCSV()
Dim enr As String
Dim Texte As Variant
Set f = ThisWorkbook.Sheets("Feuil1")
Open "C:\Essais\FichierCSV.csv" For Input As #1
li = 1
Do While Not EOF(1)
    Line Input #1, enr
    Texte = Split(enr, ";")
    For i = LBound(Texte) To UBound(Texte)
        f.Cells(li, i + 1) = Texte(i)
    Next i
    li = li + 1
Loop
Close #1
End Sub
```

Nous pouvons également lire le fichier avec un tableau croisé dynamique comme nous l'avons fait avec le ficher texte précédent, il suffit de sélectionner le délimiteur point-virgule au lieu de largeur fixe.

## *Fichiers CSV à délimiteur virgule*

Les fichiers CSV au format américain sont souvent plus facile à utiliser avec Visual Basic que les fichiers avec le point-virgule comme séparateur. Ils disposent, en effet, d'instructions plus performantes pour la lecture et l'écriture, c'est-à-dire les instructions Write # et Input # qui permettent de travailler directement avec les variables des enregistrements.

Par exemple, nous pouvons écrire un fichier CSV des ventes au format américain grâce au programme suivant :

```
Sub EcrireVirgule()
Dim produit As String
Dim vendeur As String
Dim ventes As Double
Set f = ThisWorkbook.Sheets("Feuil1")
Open "C:\Essais\VirguleCSV.csv" For Output As #1
For i = 1 To 8
    produit = f.Cells(i, 1)
    vendeur = f.Cells(i, 2)
    ventes = f.Cells(i, 3)
    Write #1, produit, vendeur, ventes
Next i
Close #1
End Sub
```

En ouvrant le fichier *VirguleCSV.csv*, nous constatons que les textes apparaissent entourés de guillemets et que les nombres sont affichés avec le point comme séparateur décimal.

Nous pouvons également le lire grâce au programme suivant :

```
Sub LireVirgule()
Dim produit As String
Dim vendeur As String
Dim ventes As Double
Set f = ThisWorkbook.Sheets("Feuil3")
Open "C:\Essais\VirguleCSV.csv" For Input As #1
i = 1
Do While Not EOF(1)
    Input #1, produit, vendeur, ventes
    f.Cells(i, 1) = produit
    f.Cells(i, 2) = vendeur
    f.Cells(i, 3) = ventes
    i = i + 1
Loop
Close #1
End Sub
```

Nous constatons que, même s'ils sont stockés avec le point comme séparateur décimal, les nombres s'affichent dans la feuille de calcul avec des virgules comme séparateur décimal.

La contrepartie négative est que nous ne pouvons pas lire directement ces fichiers avec un tableau croisé dynamique.

## Regrouper des fichiers CSV

Il est souvent intéressant, lorsqu'on travaille avec des données issues de plusieurs sources, de pouvoir regrouper en un seul plusieurs fichiers CSV. Cela est particulièrement facile lorsque les fichiers ont la même structure. Supposons, par exemple, que nous cherchions à regrouper les trois fichiers Fich1.csv, Fich2.csv et Fich3.csv structurés de la même manière.

Deux cas sont à considérer selon que la première ligne des fichiers contient ou non les titres des colonnes. Le second cas étant le plus simple, nous commencerons par lui.

Le programme ci-dessous permet de créer un fichier Fichtout.csv qui regroupe les trois fichiers CSV qui ne contiennent pas les titres des colonnes en première ligne.

```
Sub Agreger()
Dim enr as String
Open "C:\Essais\Fichtout.csv" For Output As #4
Open "C:\Essais\Fich1.csv" For Input As #1
Open "C:\Essais\Fich2.csv" For Input As #2
Open "C:\Essais\Fich3.csv" For Input As #3
For i=1 to 3
    Do while Not EOF(i)
       Line Input #i, enr
       Print #4, enr
    Loop
    Close #i
Next i
Close #4
End Sub
```

Si les fichiers contiennent en première ligne les titres des colonnes, il faut commencer par introduire une ligne de titres dans le fichier Fichtout.csv puis, pour les trois autres fichiers, lire la première ligne sans la copier dans le fichier Fichtout.csv. C'est que montre le programme suivant :

```
Sub Agreger()
Dim enr as String
Open "C:\Essais\Fichtout.csv" For Output As #4
Open "C:\Essais\Fich1.csv" For Input As #1
Open "C:\Essais\Fich2.csv" For Input As #2
Open "C:\Essais\Fich3.csv" For Input As #3
Print #4, "Produit;Vendeur;Ventes"
For i=1 to 3
   Line Input #i, enr
    Do while Not EOF(i)
       Line Input #i, enr
       Print #4, enr
    Loop
    Close #i
Next i
Close #4
End Sub
```

## *Regrouper tous les fichiers CSV d'un dossier*

On peut également vouloir regrouper tous les fichiers CSV contenus dans un dossier. Supposons, par exemple, que le dossier dans lequel se trouve le classeur Excel possède un sous-dossier nommé *FichiersCSV* où sont placés les fichiers CSV à regrouper.

Supposons que ces fichiers ont la même structure et qu'ils n'ont pas de titres en première ligne. Le programme suivant réalise le regroupement :

```
Sub Agreger()
Dim enr as String
Rep = ThisWorkbook.Path
Dossier = Rep & "\FichiersCSV\"
fichier = Dir(Dossier)
Open Rep & "\Fichtout.csv" For Output As #1
i = 2
Do While fichier <> ""
    If Right(fichier, 4) = ".csv" Then
        fic = Dossier & fichier
        Open fic for input as #i
        Do while Not EOF(i)
            Line Input #i, enr
            Print #1, enr
        Loop
        Close#i
        i = i + 1
    End If
    fichier = Dir
Loop
Close #1
End Sub
```

Dans ce programme, nous avons utilisé la fonction *Path* qui renvoie l'adresse du dossier contenant le classeur. Puis nous avons utilisé Dir(Dossier) qui renvoie le nom du premier fichier contenu dans le dossier *Dossier*. Enfin, à la fin de la boucle, *Dir* contient le nom du fichier suivant.

## Renommer, supprimer et modifier des fichiers

Il peut être utile de renommer et de supprimer des fichiers. On utilise pour cela les instructions Name As et Kill.

Par exemple, supposons que nous voulions multiplier par 2 toutes les ventes dans le fichier FichierCSV.csv. Nous ne pouvons pas le faire directement et nous allons donc créer un fichier temporaire Temp.csv.

Nous allons lire le fichier FichierCSV.csv, extraire les ventes de chaque enregistrement, les multiplier par 2, recréer un nouvel enregistrement que nous enverrons dans le fichier Temp.csv. Nous détruirons ensuite le fichier FichierCSV.csv et renommerons le fichier Temp.csv en FichierCSV.csv. Le programme pour cela sera le suivant :

```
Sub DoublerVentes()
Dim enr As String
Dim Texte As Variant
Open "C:\Essais\FichierCSV.csv" For Input As #1
Open "C:\Essais\Temp.csv" For Output As #2
Do While Not EOF(1)
    Line Input #1, enr
    Texte = Split(enr, ";")
    Print #2, Texte(0) & ";" & Texte(1) & ";" & 2 * Texte(2)
Loop
Close #1
Close #2
Kill "C:\Essais\FichierCSV.csv"
Name "C:\Essais\Temp.csv" As "C:\Essais\FichierCSV.csv"
End Sub
```

Nous pouvons vérifier avec le bloc-notes que les ventes ont bien été doublées dans le fichier FichierCSV.csv.

Nous pouvons utiliser la même méthode pour modifier un seul enregistrement. Par exemple, on peut fixer à 5000 la valeur des ventes de robes de Dupond avec le programme suivant :

```
Sub ModifVentes()
Dim enr As String
Dim Texte As Variant
Open "C:\Essais\FichierCSV.csv" For Input As #1
Open "C:\Essais\Temp.csv" For Output As #2
Do While Not EOF(1)
    Line Input #1, enr
    Texte = Split(enr, ";")
    If Texte(0) = "Robes" And Texte(1) = "Dupond" Then Texte(2) =
5000
    Print #2, Texte(0) & ";" & Texte(1) & ";" & Texte(2)
Loop
Close #1
Close #2
Kill "C:\Essais\FichierCSV.csv"
Name "C:\Essais\Temp.csv" As "C:\Essais\FichierCSV.csv"
End Sub
```

On peut également suprimer un enregistrement, par exemple celui des robes de Dupond, en remplaçant dans le programme précédent la boucle par :

```
Do While Not EOF(1)
    Line Input #1, enr
    Texte = Split(enr, ";")
```

```
    If Not (Texte(0) = "Robes" And Texte(1) = "Dupond") Then
       Print #2, Texte(0) & ";" & Texte(1) & ";" & Texte(2)
    End If
Loop
```

Pour des fichiers de taille moyenne, cette procédure est généralement rapide, pour de très gros fichiers il est généralement préférable de les décomposer en plusieurs fichiers qui seront regroupés au moment de l'analyse. Dans notre exemple, il serait possible d'avoir un fichier par client ou par produit.

## Les fichiers à accès direct

Les fichiers texte, y compris les fichiers CSV, sont de fichiers séquentiels, c'est-à-dire que tous les enregistrements sont placés les uns à la suite des autres et qu'il n'est pas possible d'accéder à un enregistrement sans lire tous ceux qui le précèdent. Lorsque les fichiers sont volumineux, cela implique qu'il peut être long d'accéder à un enregistrement particulier.

Aussi, lorsqu'il est important de pouvoir accéder rapidement à un enregistrement particulier, il est souvent préférable d'utiliser des fichiers à accès direct.

Dans un fichier à accès direct, chaque enregistrement possède un numéro et il est possible de le lire directement en l'appelant par son numéro. Tous les enregistrements doivent avoir la même longueur.

### *Ouverture d'un fichier à accès direct*

Un fichier accès direct s'ouvre comme un fichier texte en utilisant l'instruction *Open* mais en utilisant l'option *Random* :

```
Open fich For Random As #n Len = Longueur
```

Où fich désigne le nom complet du fichier, n un nombre compris entre 1 et 511 et *Longueur* la longueur de chaque enregistrement qui doit être calculée à partir de la structure des enregistrements.

Un fichier à accès direct ouvert de cette manière pourra être utilisé à la fois pour la lecture et l'écriture.

Comme les fichiers texte, un ficher à accès direct doit être fermé après son utilisation par l'instruction *Close #n*, où n est le numéro utilisé pour l'ouverture du fichier.

## *Ecrire dans un fichier à accès direct*

Pour écrire des données dans un fichier à accès direct, il faut tout d'abord décrire la structure des enregistrements puis indiquer leur longueur. Ensuite, il faut définir l'enregistrement et lui attribuer un numéro.

Pour décrire la structure des enregistrements, il faut utiliser l'instruction *Type*. Cette instruction doit être placée au tout début du module où se situent les programmes et elle est valable pour l'ensemble des programmes. Dans un programme associé à un contrôle ActiveX, il faut toutefois préciser l'option *Private*. Un exemple de définition est la suivante :

```
Private Type enreg
jour As Date
prod As String * 12
vers As String * 1
nvers As Long
ere(1 To 25, 1 To 7) As Double
End Type
```

Dans le programme utilisé pour écrire un enregistrement, nous devons d'abord le déclarer par l'instruction *Dim*. Par exemple :

```
Dim unERE As enreg
```

Pour écrire un enregistrement, il faut utiliser la fonction *Put*. Sa syntaxe est la suivante :

```
Put #n, num, unEnreg
```

Où n désigne le numéro du fichier à accès direct, num le numéro de l'enregistrement et unEnreg la variable contenant les données à saisir.

Reprenons notre exemple précédent des ventes en le présentant différemment, pour changer. Les données sont les suivantes :

| | A | B | C | D |
|---|---|---|---|---|
| 1 | Code produit | Produit | Martin | Dupond |
| 2 | 1 | Robes | 1234,12 | 3421,12 |
| 3 | 2 | Manteaux | 675,32 | 2541,34 |
| 4 | 3 | Pantalons | 3 455,64 | 2365,15 |
| 5 | 4 | Chemises | 2145,12 | 3215,37 |

Le programme suivant crée un fichier à accès direct où chaque produit correspond à un enregistrement :

```
Type enreg
produit As String * 20
ventes(1 To 2) As Double
End Type
-----------------------------------------------------------------
Sub EcrireAccesDirect()
Dim unProduit As enreg
Set f = ThisWorkbook.Sheets("Données")
Open "C:\Essais\FichierDirect" For Random As #1 Len = Len(unProduit)
For i = 1 To 4
    With unProduit
       .produit = f.Cells(i + 1, 2)
       .ventes(1) = f.Cells(i + 1, 3)
       .ventes(2) = f.Cells(i + 1, 4)
    End With
    Put #1, i, unProduit
Next i
Close #1
End Sub
```

Si l'on ouvre le fichier *FichierDirect* avec le Bloc-notes nous ne pouvons lire que les produits car ce n'est pas un fichier texte.

## *Lecture d'un enregistrement d'un fichier direct*

Pour lire un enregistrement spécifique d'un fichier à accès direct, nous devons utiliser la commande *Get* dont la syntaxe est la suivante :

```
Get #n, num, unEnreg
```

Où n est l'entier correspondant au fichier ouvert, num le numéro de l'enregistrement et unEnreg la variable de type correspondant aux enregistrements qui recevra l'enregistrement numéro *num*.

Par exemple, le programme suivant lit le troisième enregistrement correspondant aux pantalons :

```
Sub LireAccesDirect()
Dim unProduit As enreg
Set f = ThisWorkbook.Sheets("Données")
Open "C:\Essais\FichierDirect" For Random As #1 Len = Len(unProduit)
Get #1, 3, unProduit
With unProduit
    f.Cells(10, 2) = .produit
    f.Cells(10, 3) = .ventes(1)
    f.Cells(10, 4) = .ventes(2)
End With
Close #1
End Sub
```

**Auteur : Francis Malherbe**