

Les Fonctions Date/Heure

par Maxence Hubiche ([site](#)) ([Blog](#))

Date de publication : 04/01/2008

Dernière mise à jour :

"Les fiches VBA" sont une série de petits tutoriels rapides regroupant les informations utiles sur un sujet donné. Elles concernent le langage VBA dans son ensemble et ne sont pas spécifiques à un logiciel donné. Vous y trouverez donc des informations valables pour Access, Excel, Word, Outlook, PowerPoint, ... Bonne lecture !

- I - Qu'abordons-nous ici ?
- II - Généralité sur les dates
- III - Les fonctions
 - III-A - Les fonctions de base
 - III-B - Les fonctions d'extraction simples
 - III-C - Les fonctions de reconstruction
 - III-C-1 - Exemples
 - III-D - Les fonctions sur intervalles
 - III-D-1 - Les intervalles
 - III-D-2 - Exemples
 - III-E - Les fonctions de conversion et d'information
 - III-E-1 - A savoir...
 - III-E-1-a - Dans le module "Information"
 - III-E-1-b - Dans le module "Conversion"
 - III-E-1-c - Dans le module "Strings"
- IV - Information spécifique Access
- V - Bonus ! Fonctions clé en main
- VI - Remerciements

I - Qu'abordons-nous ici ?

Voici une fiche relative aux fonctions disponibles dans la bibliothèque VBA. Elle concerne exclusivement les fonctions de Date/Heure (module DateTime)

Vous noterez que certaines fonction (Date, Time, ...) peuvent parfois être suffixées d'un \$. Ce symbole indique que la fonction renvoie l'information sous la forme d'une valeur de type String (Chaîne de caractères).

II - Généralité sur les dates

Les dates sont stockées dans un **numérique** à virgule flottante sur **8 octets (64 bits IEEE754 (1))**, dont la partie entière représente la date et la partie décimale l'heure.

La date de référence est le 1/1/1900 qui a pour valeur 2 (2). En effet, 1900 n'étant pas une année bissextile et donc étant une année à laquelle il manque un jour, il suffisait de rajouter un jour pour récupérer le décalage ainsi induit.

- Un **Jour** = 1
- Une **Heure** = $1/24 = 0.04166666666666666666666666666667$
- Une **Minute** = $1/1440 = 0.0006944444444444444444444444444444$
- Une **seconde** = $1/86400 = 0.000011574074074074074074074074074$

Dans le calendrier grégorien, il est possible de stocker toutes les dates comprises entre le 1/1/100 et le 31/12/9999

III - Les fonctions

III-A - Les fonctions de base

Ces fonctions sont les fonctions fondamentales. Elles vous permettent de récupérer des informations système. Elles n'ont pas d'argument. Il suffit d'utiliser leur nom pour récupérer l'information qu'elles dispensent.

Elles sont symbolisées par le pictogramme des propriétés (3) et permettent comme toutes fonction, de récupérer un résultat, mais certaines (**Date**, **Time**, **Calendar**) permettent également de modifier les paramètres de Windows. Nous ne nous intéresserons cependant ici qu'à l'aspect Fonction (Lecture de l'information).

- **Calendar** : La fonction renvoie une valeur de type **Long** correspondant à l'une des constantes de l'énumération vbCalendar (vbCalGreg=0 pour le calendrier Grégorien et vbCalHijri=1 pour le calendrier Hijri (4))
- **Date** (5) : La fonction renvoie **la date système en cours** sous la forme d'un **Variant** de sous-type **Date**.
- **Now** : La fonction renvoie **la date et l'heure système en cours** sous la forme d'un **Variant** de sous-type **Date**.
- **Time** : La fonction renvoie **l'heure système en cours** sous la forme d'un **Variant** de sous-type **Date**.
- **Timer** : La fonction renvoie le **nombre de secondes** écoulées depuis minuit, *au centième de seconde près*. Type de données renvoyées : **Single**

III-B - Les fonctions d'extraction simples

Ces fonctions sont des fonctions qui n'attendent qu'un seul argument qui peut être n'importe quelle expression de type Variant, expression numérique, expression de chaîne ou toute combinaison pouvant représenter une date.

Elles vont pouvoir renvoyer une partie de la date ou de l'heure passée en paramètre.

- **Year** : La fonction renvoie un **Integer** correspondant au **numéro de l'année** (de 100 à 9999) de la date passée en paramètre
- **Month** : La fonction renvoie un **Integer** correspondant au **numéro du mois** (de 1 à 12) de la date passée en paramètre
- **Day** : La fonction renvoie un **Integer** correspondant au **jour du mois** (de 1 à 31) de la date passée en paramètre
- **WeekDay** : La fonction renvoie un **Integer** correspondant au **jour de la semaine** (de 1 à 7) de la date passée en paramètre. Cette fonction contient un deuxième argument, facultatif, qui va permettre de déterminer le premier jour de la semaine. Cet argument contient une valeur qui est une constante de l'énumération **vbDayOfWeek** (vbMonday, vbTuesday, vbWednesday, vbThursday, vbFriday, vbSaturday, vbSunday). Par défaut, le premier jour de la semaine est vbSunday (6) , ce qui ne correspond pas au calendrier français, mais au calendrier US. Le retour de cette fonction est l'une des constantes de l'énumération vbDayOfWeek.
- **Hour** : la fonction renvoie **Integer** correspondant à la **partie de l'heure** (de 0 à 23) de la date passée en paramètre
- **Minute** : la fonction renvoie **Integer** correspondant à la **partie des minutes** (de 0 à 59) de la date passée en paramètre
- **Second** : la fonction renvoie **Integer** correspondant à la **partie des secondes** (de 0 à 59) de la date passée en paramètre

III-C - Les fonctions de reconstruction

Dans cette catégorie, j'ai mis deux fonctions qui ont une structure identique. Elles ont toutes les deux 3 arguments qui représentent des périodes de temps.

- **DateSerial** : La fonction renvoie un **Variant (Date)** par reconstruction d'une date, en fonction d'une valeur d'année, de mois et de jour. Ces trois valeurs sont renseignées dans trois arguments nommés respectivement *Year*, *Month* et *Day*.
- **TimeSerial** : La fonction renvoie un **Variant (Date)** par reconstruction d'un temps, en fonction d'une valeur d'heure, de minutes et de secondes. Ces trois valeurs sont renseignées dans trois arguments nommés respectivement *Hour*, *Month* et *Day*.

III-C-1 - Exemples

Premier exemple : La fonction ci-après vous permet de retrouver la date correspondant au **premier jour du mois en cours**

```
DateSerial(Year(Date()), Month(Date()), 1)
```

Deuxième exemple : La fonction ci-après vous permet de retrouver la date correspondant au **dernier jour du mois précédent**. Puisque les dates sont des nombres, il suffit de retirer 1 (valeur d'un jour) au premier jour du mois en cours.

```
DateSerial(Year(Date()), Month(Date()), 1) - 1
```

On aurait aussi pu l'écrire ainsi

```
DateSerial(Year(Date()), Month(Date()), 0)
```

Ces fonctions sont capables de passer les limites des cycles. Ainsi, les exemples précédents fonctionnent même lorsqu'on est en janvier. Si jamais l'argument month était défini à 13, la fonction **DateSerial** reconstruirait une date pour le mois de janvier de l'année suivant celle indiquée dans l'argument Year.

Troisième exemple : Déterminer l'**intervalle entre le premier jour d'il y a 3 mois et le dernier jour du mois précédent**. Avec ces syntaxes, nous aurons toujours des informations relatives aux trois derniers mois, de mois à mois.

```
Between DateSerial(Year(Date()), Month(Date()) - 3, 1) And DateSerial(Year(Date()), Month(Date()), 0)
```

On aurait aussi pu l'écrire ainsi

```
<= DateSerial(Year(Date()), Month(Date()) - 3, 1) And >= DateSerial(Year(Date()), Month(Date()), 0)
```

Quatrième et dernier exemple : Conservons le principe des trois mois, mais il s'agit cette fois de **3 mois roulants**, c'est à dire de date à date. Nous pourrions faire ceci :

```
Between DateSerial(Year(Date()), Month(Date()) - 3, Day(Date())) And DateSerial(Year(Date()), Month(Date()), Day(Date()))
```

On aurait aussi pu l'écrire ainsi

```
<= DateSerial(Year(Date()), Month(Date()) -3, Day(Date())) And >= DateSerial(Year(Date()),  
Month(Date()), Day(Date()))
```

Les exemples cités ci-dessus n'utilisent que la fonction **DateSerial**, cependant d'autres exemples auraient pu être faits avec **TimeSerial** en remplaçant les arguments Year, Month et Day par Hour, Minute et Second.

III-D - Les fonctions sur intervalles

Les fonctions sur intervalles sont des fonctions qui travaillent sur les dates, en fonction de divers arguments. L'un d'entre eux s'appelle *interval*. Cet argument reçoit des codes qui déterminent la partie de date qui servira dans la réponse de la fonction d'intervalles de référence. Pour la liste des codes possibles, voir la partie sur [les codes interval](#)

- **DateAdd** : La fonction renvoie un [Variant \(Date\)](#) correspondant à une date de référence décalée d'un nombre d'intervalles. La fonction possède 3 arguments :
 - * [Interval](#) : L'une des codes d'interval mentionnés plus bas
 - * [Number](#) : Nombre d'*Interval* à ajouter à la *Date*. Cet argument peut être un nombre positif ou négatif
 - * [Date](#) : La date de référence, à laquelle il faut ajouter un *Number d'Interval*
- **DateDiff** : La fonction renvoie un [Variant\(Long\)](#) qui correspond au nombre d'intervalles qu'il y a entre deux dates. Cette fonction possède 3 arguments :
 - * [Interval](#) : L'une des codes d'interval mentionnés plus bas
 - * [Date1](#) : L'une des deux dates de l'interval à mesurer. Il convient de mettre ici une date antérieure ou égale à *Date2*. Dans le cas contraire, la fonction retournerait une valeur négative.
 - * [Date2](#) : L'une des deux dates de l'interval à mesurer. Il convient de mettre ici une date postérieure ou égale à *Date1*. Dans le cas contraire, la fonction retournerait une valeur négative.
 - * [FirstDayOfWeek](#) : Il s'agit d'une des constantes de l'énumération vbDayOfWeek (7) qui permettra de déterminer le premier jour de la semaine. Par défaut, sa valeur est vbSunday (système US)
 - * [FirstWeekOfYear](#) : Il s'agit d'une des constantes de l'énumération vbFirstWeekOfYear (8) qui permettra de déterminer la première semaine de l'année. Par défaut, la valeur est vbFirstJan1 (semaine du premier janvier). Le calendrier français est régi par la norme ISO 8601:2000 qui déclare que la première semaine de l'année est la première semaine contenant le jeudi. Une configuration pour le calendrier français reviendrait donc à mettre *vbFirstDayOfWeek* à *vbMonday* et *vbFirstWeekOfYear* à *vbFirstFourDays*
- **DatePart** : La fonction renvoie un [Variant \(Integer\)](#) qui correspond au numéro de l'*Interval*, dans la *Date* passée en paramètre.
 - * [Interval](#) : L'une des codes d'interval mentionnés plus bas
 - * [Date](#) : La date dont nous souhaitons récupérer le numéro de l'*Interval*.
 - * [FirstDayOfWeek](#) : Il s'agit d'une des constantes de l'énumération vbDayOfWeek qui permettra de déterminer le premier jour de la semaine. Par défaut, sa valeur est vbSunday (système US)

* **FirstWeekOfYear** : Il s'agit d'une des constantes de l'énumération `vbFirstWeekOfYear` qui permettra de déterminer la première semaine de l'année. Par défaut, la valeur est `vbFirstJan1` (semaine du premier janvier). Le calendrier français est régi par la norme ISO 8601:2000 qui déclare que la première semaine de l'année est la première semaine contenant le jeudi. Une configuration pour le calendrier français reviendrait donc à mettre `vbFirstDayOfWeek` à `vbMonday` et `vbFirstWeekOfYear` à `vbFirstFourDays`

III-D-1 - Les intervalles

Les intervalles sont des codes de type **String**.

US	Explication	FR
"yyyy"	Année (de 100 à 9999)	"aaaa"
"y"	Jour de l'année (de 1 à 366)	"a"
"m"	Mois (de 1 à 12)	"m"
"q"	Trimestre (de 1 à 4)	"t"
"ww"	Semaine (de 1 à 53)	"ee"
"w"	Jour de la semaine (de 1 à 7)	"e"
"d"	Jour du mois (de 1 à 31)	"j"
"h"	Heure (de 0 à 23)	"h"
"n"	Minute (de 0 à 59)	"n"
"s"	Seconde (de 0 à 59)	"s"

Les codes de la colonne US sont à utiliser dans le code SQL des requêtes et dans les modules et modules de classe. L'usage des codes en français est plus restreint. Il ne convient de les utiliser que dans les champs calculés des formulaires, états, et grille QBE d'Access, bref, à chaque fois que vous souhaitez utiliser la fonction dans une interface en français.

III-D-2 - Exemples

Premier exemple : Peut-on connaître la date précise d'il y a 3 mois ? Il suffit d'ajouter un nombre de mois négatif (-3 "m") à la date en cours (Date())

```
DateAdd("m", -3, Date())
```

Deuxième exemple : Comment calculer le nombre de trimestres entre une date (d'embauche, stockée dans une variable `datDateEmbauche`) et une autre (l'actuelle) ?

```
DateDiff("q", datDateEmbauche, Date())
```

Troisième exemple : Quelle semaine sommes-nous, dans le calendrier français ?

```
DatePart("ww", Date(), vbMonday, vbFirstFourDays)
```

III-E - Les fonctions de conversion et d'information

Il existe encore deux fonctions dans le module DateTime qui sont des fonctions de conversion. Elles convertissent une date ou une heure passées en paramètre String en une valeur de type Date. Leur syntaxe est très simple :

- **DateValue** : La fonction renvoie un **Variant (Date)** qui correspond au numéro de série de la *Date* passée en paramètre.
 - * Date : Une chaîne correspondant à une date valide en fonction des paramètres Windows. Cette expression peut donc intégrer les séparateurs de date (/) qui y sont définis. La plupart du temps, ces solutions seront acceptées : "31/7/1960", "31/07/1960", "31/7/60", "31 juillet 1960", "31-juil-1960" et "31 juil 1960"
- **TimeValue** : La fonction renvoie un **Variant (Date)** qui correspond au numéro de série de la *Date* passée en paramètre. Cette fonction ne renvoie que la partie horaire du numéro de série généré, ne prenant pas en compte la partie date.
 - * Date : Une chaîne correspondant à une heure valide en fonction des paramètres Windows, que ce soit en cycle de 12 ou de 24 heures. Les valeurs "08:00PM" et "20:00" sont des valeurs autorisées pour cet argument.

III-E-1 - A savoir...

Je vais maintenant vous montrer quelques fonctions qui ne font pas partie du module DateTime, cependant, comme ces fonctions sont très étroitement liées aux notions de Date et Heures, je pense qu'il est important de faire le topo Date/Heure jusqu'au bout.

Trois autres modules sont intéressants : Les modules **Information**, **Conversion** et enfin le module **Strings**.

III-E-1-a - Dans le module "Information"

Nous y trouvons une fonction très importante. La seule fonction qui nous permet de nous assurer qu'une information passée en paramètre de la fonction est bien interprétable comme étant une Date ou peut être convertie en Date. En voici la syntaxe :

- **IsDate** : La fonction renvoie un **Boolean** indiquant si la valeur passée en paramètre peut être convertie en Date.
 - * Expression : Toute Expression de type Variant qui peut être interprété comme étant une date (pour renvoyer **True**)

III-E-1-b - Dans le module "Conversion"

Ce module, qui contient les fonctions relatives au transtypage (Cast), doit contenir au moins une fonction pour 'caster' en Date.

- **CDate** : La fonction renvoie une **Date** suite à la conversion de la valeur passée en paramètre.
 - * Expression : Toute Expression de type Variant qui peut être interprété comme étant une date

- **CVDate** : Cette fonction travaille sur le même principe que CDate, la seule différence étant qu'elle ne renvoie pas une valeur de type Date, mais une valeur de type **Variant(Date)**. Eu égard à cette différence, il n'est pas conseillé d'utiliser CVDate, mais plutôt CDate, qui renvoie un véritable type Date. Cette fonction n'est conservée qu'à des fins de compatibilité avec les versions précédentes.

III-E-1-c - Dans le module "Strings"

Ce module est dédié à la gestion des chaînes de caractères. Mais il existe une fonction qui, bien que renvoyant un String, peut travailler sur des données de type Date

- **Format** : La fonction renvoie un **Variant(String)** qui correspond au formatage d'une expression.
 - * **Expression** : Toute expression valide, dit l'aide. Dans notre cas, toute Date valide.
 - * **Format** : un format valide. Ce format peut utiliser les codes d'intervalle, ainsi que certaines combinaisons, telles que "dddd" pour le nom du jour complet, "ddd" pour le nom du jour en abrégé, "dd" pour le numéro du jour du mois sur deux positions et "d" pour le numéro du jour du mois, sans 0 significatif. De même pour les mois (avec "mmmm", "mmm", "mm" ou "m") et les années (avec "yyyy" ou "yy").
 - * **FirstDayOfWeek** : Il s'agit d'une des constantes de l'énumération vbDayOfWeek qui permettra de déterminer le premier jour de la semaine. Par défaut, sa valeur est vbSunday (système US)
 - * **FirstWeekOfYear** : Il s'agit d'une des constantes de l'énumération vbFirstWeekOfYear qui permettra de déterminer la première semaine de l'année. Par défaut, la valeur est vbFirstJan1 (semaine du premier janvier). Le calendrier français est régi par la norme ISO 8601:2000 qui déclare que la première semaine de l'année est la première semaine contenant le jeudi. Une configuration pour le calendrier français reviendrait donc à mettre *vbFirstDayOfWeek* à *vbMonday* et *vbFirstWeekOfYear* à *vbFirstFourDays*

Un exemple de la fonction FORMAT dans une chaîne SQL

```
SELECT CdeID, CdeDate, CdeDateEnvoi
FROM tblCommandes
WHERE CdeDate=Format(Date(), "mm/dd/yyyy")
```

IV - Information spécifique Access

Dans Access, il est possible d'utiliser ces fonctions dans les états, formulaires et même dans la grille QBE. La particularité d'Access est de fournir ces fonctions en Français dans un IDE français. Voici donc les correspondances FR-US de ces fonctions de Date. A noter que, dans la grille QBE, vous pouvez utiliser indifféremment les noms US et FR.

FR	US
AjDate()	DateAdd()
Année()	Year()
CDate	CDate
CVDate	CVDate
Date()	Date()
DiffDate()	DateDiff()
EstDate	IsDate
Format	Format
Heure()	Hour()
Jour()	Day()
JourSem()	WeekDay()
Maintenant()	Now()
Minute()	Minute()
Minuterie	Timer
Mois()	Month()
PartDate()	DatePart()
Seconde()	Second()
SérieDate()	DateSerial()
SérieHeure()	TimeSerial()
Temps()	Time()
ValDate()	DateValue()
VHeure()	TimeValue()

V - Bonus ! Fonctions clé en main

Ci-dessous, je vous propose une série de fonctions que vous pourrez réutiliser librement et qui ne sont livrées qu'à titre d'exemples. Elles sont utilisables aussi bien dans vos requêtes, formulaires, états que procédures. Ces fonctions permettent de passer outre les limitations relatives au calendrier US utilisé par défaut et vous affranchissent du paramétrage des arguments FirstWeek et FirstWeekDay. Elles apportent également quelques notions amusantes (par exemple, le dernier jour du mois d'une date donnée)

```
'-----  
' Copyright : Ce code est librement ditribuable, copiable et imprimable, sous la seule  
'           contrainte de laisser visible la totalité des commentaires identifiant  
'           l'auteur de ce code, ses coordonnées et ce copyright et ce, sans  
'           limitation de durée dans le temps.  
'-----  
' Module      : modDateTimeFonctions  
' Date        : vendredi 21 décembre 2007 13:53  
' Auteur      : Maxence Hubiche (mhubiche@club-internet.fr - 06.18.61.14.35)  
'-----  
Option Explicit  
  
Public Function Semaine(LaDate As Variant) As Variant  
'-----  
' Procédure : Semaine  
' Date      : vendredi 21 décembre 2007 13:56  
' Auteur    : Maxence Hubiche (mhubiche@club-internet.fr - 06.18.61.14.35)  
' Objet     : Renvoie le numéro de la semaine pour le calendrier français  
' Spec     : Cette fonction empêche le débordement en semaine 53, en mettant les jours  
'           de la semaine 53 en semaine 1  
' Retour    : Renvoie un Byte (n° de la semaine) ou Null si l'argument n'était pas une  
'           date  
'-----  
  
Dim bytTemp As Byte  
If IsDate(LaDate) Then  
    bytTemp = CByte(DatePart("ww", LaDate, vbMonday, vbFirstFourDays)) Mod 53  
    If bytTemp = 0 Then bytTemp = 1  
    Semaine = bytTemp  
Else  
    Semaine = Null  
End If  
End Function  
  
Public Function Trimestre(LaDate As Variant)  
'-----  
' Procédure : Trimestre  
' Date      : vendredi 21 décembre 2007 17:18  
' Auteur    : Maxence Hubiche (mhubiche@club-internet.fr - 06.18.61.14.35)  
' Objet     : La fonction renvoie le trimestre de la date.  
' Spec     : si des journées de décembre sont en semaine 1 de l'année suivante, alors  
'           ces dates vont dans l'année suivante en tant qu'éléments du trimestre 1  
' Retour    : Renvoie un Byte (n° du trimestre) ou Null si l'argument n'était pas une  
'           date  
'-----  
  
Dim bytTemp As Byte  
If IsDate(LaDate) Then  
    bytTemp = Semaine(LaDate)  
    If bytTemp = 1 Then  
        Trimestre = 1  
    Else  
        Trimestre = DatePart("q", LaDate, vbMonday, vbFirstFourDays)  
    End If  
Else  
    Trimestre = Null
```

```
End If
End Function

Function DernierJour(LaDate As Variant) As Variant
'-----
' Procedure : DernierJour
' Date      : vendredi 21 décembre 2007 17:29
' Auteur    : Maxence Hubiche (mhubiche@club-internet.fr - 06.18.61.14.35)
' Objet     : Renvoi le dernier N° du jour du mois de la date passée en argument
' Retour    : Renvoie un Byte (n° du jour) ou Null si l'argument n'était pas une
'            date
'-----
'
  If IsDate(LaDate) Then
    DernierJour = CByte(Day(DateSerial(Year(LaDate), Month(LaDate) + 1, 0)))
  Else
    DernierJour = Null
  End If
End Function
```

VI - Remerciements

Je tiens à remercier le travail d'équipe de tous les bénévoles qui participent à ce site et, dans le cadre des observations et corrections apportées à ce tutoriel, mes remerciements vont tout particulièrement à

Correction

- [Starec](#)
- [Argyronet](#)

Commentaires

- [Heureux-Oli](#)

1 : L'IEEE 754 est un standard qui détermine la représentation des nombres à virgule flottante. Pour les nombres de 64bits, il décrit 1 bit de signe, 11 bits d'exposant et 52 bits de mantisse.

2 : Ces dates de références sont données pour le système Windows sur PC. Sur Macintosh, la date de référence est le 1/1/1904, qui vaut 1

3 : Sans entrer dans le détail, le module DateTime est en fait un module de classe static. Un module de classe static est un module ne s'instanciant pas, c'est à dire qu'on ne peut pas créer d'objet de type DateTime. Cependant, l'objet DateTime existe. Ses méthodes et propriétés sont accessibles directement dans le code, sans avoir à passer par l'objet DateTime. Ainsi, par exemple, même s'il est possible d'écrire VBA.DateTime.Date, la simple mention de la fonction Date par son nom suffit. Maintenant que ce point est vu, nous comprenons pourquoi certaines fonctions ont un pictogramme de propriété : ce sont des propriétés, en lecture-écriture de l'objet DateTime. Les propriétés en lecture sont considérées comme des fonctions. Les propriétés en écriture sont considérées comme des méthodes. Avec ces fonctions, il est donc possible, à la fois, de demander l'information de date que de la modifier.

4 : Calendrier Hijri : Le calendrier Hijri est le calendrier musulman. Ce calendrier est un calendrier de 12 mois lunaires qui débuta le premier jour de l'égire, le 15 (époque astronomique) ou le 16 (époque civile) juillet 622

5 : Comme indiqué dans la note n°3, certaines 'fonctions' sont en fait des propriétés. **Date** est une propriété. Syntaxiquement parlant (cf. : <http://mhubiche.developpez.com/vba/fiches/syntaxes/bases>), il ne convient pas de mettre des parenthèses. Cependant, pour ne pas déroger au principe des 'fonctions', dans cette fiche, nous mettrons les parenthèses systématiquement. Ne vous étonnez pas si elles sont automatiquement supprimées après que vous les ayez inscrites.

6 : Dans la mesure où vous faites usage de cette fonction dans un module, pensez à mettre ce paramètre à vbMonday, pour rétablir le premier jour de la semaine selon un calendrier français. Il est cependant regrettable de ne pas pouvoir définir correctement ce dernier dans une requête Access

7 : Les constantes de l'énumération vbDayOfWeek : vbSunday (=1), vbMonday (=2), vbTuesday (=3), vbWednesday (=4), vbThursday (=5), vbFriday (=6), vbUseSystemDayOfWeek (=0)

8 : Les constantes de l'énumération vbFirstWeekOfDay : vbFirstJan1 (=1), vbFirstFourDays (=2), vbFirstFullWeek (=3), vbUseSystem (=0)