Descriptif des conventions typographiques du code Visual Basic

Table des matières

Ce document a pour objectif de simplifier la compréhension de l'écriture du code Visual Basic en vous montrant comment appliquer des conventions typographiques adaptées au code.

Son but, faire en sorte que votre code soit plus lisible pour vous-même et tous les membres susceptibles de le relire.

L'auteur

Jean-Philippe AMBROSINO

L'article

Publié le 22 mars 2005 - Mis à jour le 27 avril 2005

1. Avant propos

Il est fortement recommandé voir impératif d'appliquer certaines règles en matière de conventions typographiques du code de programmation et uniquement dans le but de faciliter la relecture.
Ce document est dédié à vous aider dans cette manœuvre.

1-1. Normalisation des noms

Developpez.com vous propose déjà de consulter un fichier de normalisation des noms.

Vous pouvez le télécharger en cliquant ici.
Il s'agit d'un fichier au format Excel qui récapitule la liste des types de contrôles et des variables dont vous trouverez des détails supplémentaires au sein de ce document.

1-2. Remerciements

Je tiens à remercier tout particulièrement Demco et Tofalu ainsi que toutes celles et ceux qui ont participé à la relecture de ce document en y incluant leurs remarques.

1-3. Contact

Pour tous renseignements complémentaires, veuillez me contacter directement (Argyronet) par MP.

2. Pourquoi définir une convention au niveau du code ?

La raison majeure qui oblige à respecter cette convention est fondée sur l'objectif de standardiser la structure et le style du code du programme d'un projet ou d'une application afin que vous et tout autre personne faisant partie de votre équipe soit à même de lire, analyser et comprendre plus rapidement le contenu.

Le fait d'appliquer l'écriture de son code avec de telles conventions permet d'obtenir un code source précis,

dépourvu d'ambiguïtés et surtout facile à relire.

Ces conventions sont à appliquer pour le cas d'une variable, selon sa portée et d'un objet (Table et ses champs, Formulaire et ses contrôles, Etat et ses contrôles, Module etc...) selon son type.

Cela simplifie grandement le développement en équipe du fait que des normes sont mises en place.
La relecture en est (et restera) plus aisée. Il faut noter au passage que les conventions de dénomination sont à établir au départ. Une fois le développement réalisé, il est difficile de revenir en arrière pour intervenir sur le code dans le seul but d'appliquer ces conventions.

2-1. Syntaxe des conventions

Il existe des normes et des conventions préétablies pour nommer les objets de contrôle, les variables, les procédures et les constantes et tous les objets faisant partie du code.
Ces normes sont issues de règles logiques qui permettent en un seul coup d'oeil de repérer ce à quoi on a à faire sans qu'il soit forcément nécessaire de partir à la recherche de la source de l'objet concerné.

2-1-1. Syntaxe des conventions pour les objets

Terminologie des noms d'objets

De manière générale, un objet de contrôle doit obligatoirement être renommé (par défaut un objet de contrôle porte le nom du type de contrôle suivi d'un numéro incrémenté).

Par exemple:
- Liste13
- Étiquette47
- Texte14
- Commande23

La terminologie standard d'un nom d'objet peut être scindée au minimum en **2 parties** et au maximum **4**.

La terminologie en 2 parties pour les noms d'objets se compose comme suit:
[Une étiquette]Le Nom de l'Objet

La terminologie en 3 parties pour les noms d'objets se compose comme suit:
[Le Préfixe][Une étiquette]Le Nom de l'Objet

La terminologie en 4 parties pour les noms d'objets se compose comme suit:
[Le Préfixe][Une étiquette]Le Nom de l'Objet[Le Suffixe]

Pour appliquer la terminologie d'un objet, il suffit de se référer au tableau que vous trouverez sur le site ici ou dans ce même document là.
Ce qui est délimité par des crochets est facultatif. Toutefois, cela ne rejoint plus alors l'objectif de ce document.

Définition du Préfixe
Le préfixe a pour rôle d'identifier le type de l'objet:

- - On préfixera les objets contrôles mais on apposera pas d'étiquette
- - On préfixera les objets tables, requêtes, formulaires...
- - On préfixera les objets de type modules et classes...

Quelques préfixes pour un objet de type contrôle

- - **lbl** => Label : Étiquette (Ne pas confondre avec l'étiquette de nommination)
- - **txt** => Textbox : Zone de texte
- - **cmd** => CommandButton : Bouton de commande
- - **shp** => Shape : Forme dessinée (rectangle, cercle...)
- - **cbo** => Combobox : Liste modifiable
- - **lst** => Listbox : Zone de Liste
- etc...

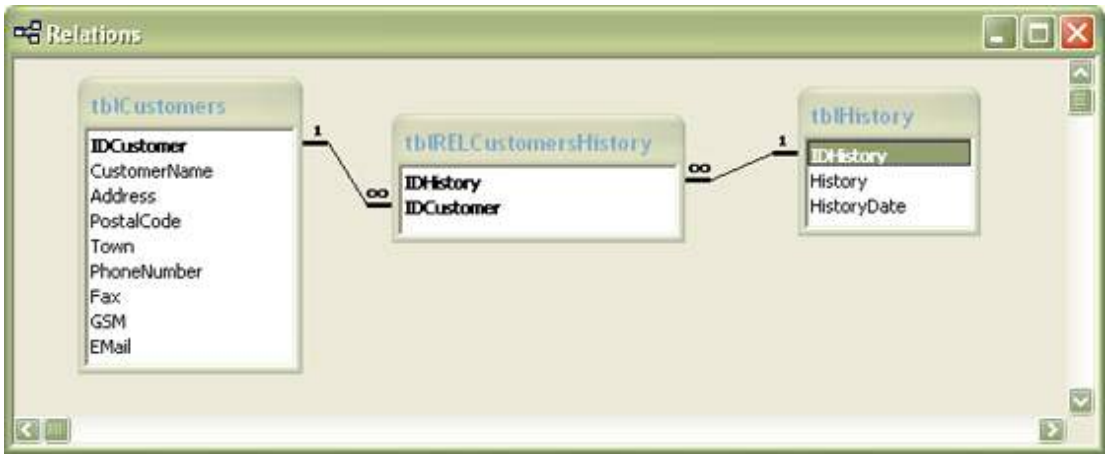
Quelques préfixes pour un objet d'application

- - **tbl** => Table : Table
- - **qry** => Query : Requête ou Vue
- - **frm** => Form : Formulaire
- - **rpt** => Report : État
- - **mcr** => Macro : Macro
- - **bas** => Module : Feuille de module
- - **cls** => Classe : Feuille de classe
- - **cmb** => CommandBar : Barre de commande
- - **mnu** => Menus : Barre de menus
- etc...

Définition de l'Étiquette pour un objet de type contrôle
Il n'est pas utile de poser une étiquette pour un contrôle car l'étiquette définit la catégorie. Or, pour un contrôle, le préfixe définissant déjà son type, cela est amplement suffisant.

Définition de l'Étiquette pour un objet d'application
Une étiquette est un identificateur de catégorie de l'objet mêlant ainsi le rôle de l'objet ou ce à quoi il est destiné.
Une étiquette n'aura pas plus de 3 ou 4 caractères et sera représentée par un abrégé conventionnel.

A) Par exemple, pour un objet de type **Table**, on peut imaginer qu'une table de relation soit située comme table intercalée entre une table principale et une table intermédiaire. Elle doit pouvoir alors être identifiée comme telle.



L'étiquette **REL** définit le fait que la table est une Table de Relation même si l'on devine par son **Nom d'Objet** que la table sollicite les clients (Customers) et l'historique de ces derniers (History).

B) Un autre exemple, relatif à un objet de type requête (Query), où l'on peut étiqueter la requête par son rôle

;
Par exemple, le nom **qrySumOrderAmountsForFrance** où **Sum** est l'étiquette qui définit qu'il s'agit d'une requête avec une clause **GROUP BY** et où le **Nom de l'Objet** informe que cette requête renvoie le total des montants des commandes pour la France.

Définition du Nom de l'Objet
Le nom de l'objet sera aussi explicite que possible tout en respectant une limite de caractères.
Il commencera toujours par un caractère alphabétique et sera (exceptionnellement) greffé de caractères de soulignement (_) **et surtout pas d'espaces**. De plus, on apposera une **MAJUSCULE** à la première lettre de chaque mot qui le constitue.

Pour les sous-requêtes, les sous-formulaires et les sous états, j'utilise personnellement le suffixe **_sub** comme indicateur.
Cela me permet de faire en sorte que les objets Parents et Enfants concernés soient rassemblés alphabétiquement ensemble dans leurs onglets respectifs et que cela soit visible rapidement.

Exemple:

Objet Parent	Objet Enfant
frmEntryNewOrder	frmEntryNewOrder_sub
qryFrmInquireCustomersAndOrders	qryFrmInquireCustomersAndOrders_sub
rptBillCustomers	rptBillCustomers_sub

Définition d'un Suffixe
Le suffixe est très optionnel et reste utile pour marquer la différence entre deux objets qui ont le même rôle, le même type mais n'attaquant ou ne retournant pas les mêmes données. Par exemple, il est fréquent de rencontrer des contrôles texte répétés pour un numéro de téléphone ou une adresse:

txtAddresse1 et txtAdresse2 où **1** et **2** sont les suffixes au même titre que PhoneNumber1 et PhoneNumber2

Un autre exemple de suffixe, plus précis cette fois, issu d'un nom d'objet qui définit la liste des codes postaux et des villes au sein d'un contrôle **ListBox** ; comme vous le savez, un contrôle **ListBox** possède un grand nombre de propriétés mais la plus importante est sa **source de contrôle** qui lui sert à renvoyer les données.
Cette source de contrôle se traduit
- soit par une clause **SELECT** inscrite directement dans la zone prévue à cette effet,
- soit par la sélection d'un objet source, en l'occurrence une requête ou une table.

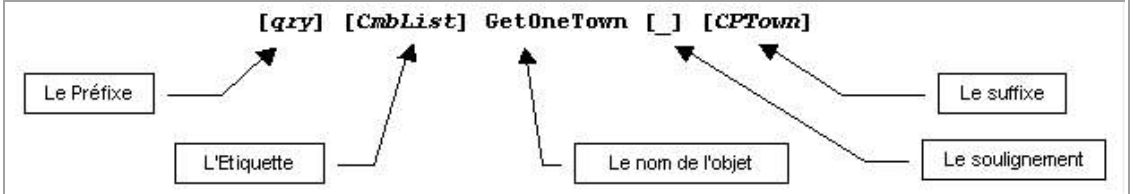
Pour cet exemple, il s'agit bien entendu d'une requête...
Ci-dessous, vous pouvez constater que selon l'intitulé du bouton **[Code postal / Ville]** ou **[Ville / Code postal]** la liste inverse et propose respectivement les codes postaux ou les villes en conséquence des caractères tapés dans le contrôle Textbox au-dessus.

Code postal - Ville	Ville - Code postal

Programmaticalement pour chacun des cas, le contrôle fait appel à deux requêtes que j'ai nommées dans un cas
qryCmbListGetOneTown_CPTown
et dans l'autre
qryCmbListGetOneTown_TownCP.

La requête source a pour objectif de renvoyer les mêmes données avec un suffixe différent, donc un ordre de colonne différent.
Vous pouvez alors remarquer que le nom que j'ai donné à ma requête est composée de **4 parties** :

Nom de l'objet Requête



Dénomination		Explications
qry	=>	Préfixe qui désigne que l'objet est une requête (abréviation de Query)
DropList	=>	Etiquette qui identifie la catégorie. Là, je sais que ma requête est dédiée à être exploitée par un contrôle de type ListBox ou ComboBox.
GetOneTown	=>	Nom de l'objet qui désigne l'objectif de la requête à savoir obtenir une ville.
_	=>	Séparateur de lisibilité (Underscore) qui a pour rôle de séparer le nom de l'objet de son suffixe.
CPTown	=>	Suffixe qui désigne que la requête renvoie les colonnes Code Postal et Ville respectivement en colonne(0) et colonne (1)

Pour résumer, vous avez pu constater que ma dénomination rien que par son nom, est très explicite et permet d'identifier rapidement le rôle de cet objet requête. On considère bien entendu qu'elle n'est pas dédiée à un contrôle nommé précis mais à tous les contrôles de type "**Liste déroulante**" susceptibles d'être présents dans n formulaires.

2-1-2. Syntaxe des conventions pour les variables

Terminologie des noms de variables

De manière générale, une variable est nommée en deux parties
La terminologie standard d'un nom de variable peut être scindée au minimum en **2 parties** et au maximum **4** tout comme les objets.
On reprendra alors la composition de la syntaxe comme suit :

La terminologie en 2 parties pour les noms d'objets se compose comme suit :
[Une étiquette]Le Nom de la Variable

La terminologie en 3 parties pour les noms d'objets se compose comme suit :
[Le Préfixe][Une étiquette]Le Nom de la Variable

La terminologie en 4 parties pour les noms d'objets se compose comme suit :
[Le Préfixe][Une étiquette]Le Nom de la Variable[Le Suffixe]

Pour bien comprendre la terminologie d'une variable, il est conseillé de lire la section suivante qui relate de leurs portées. Ce qui est délimité par des crochets est facultatif. Toutefois, cela ne rejoint plus alors l'objectif de ce document.

Définition du Préfixe
Le préfixe a pour rôle d'identifier la portée de la variable :

- - On ne préfixera pas les variables de procédures
- - On préfixera les variables de modules selon une convention
- - On préfixera les variables publiques selon une autre...

Définition d'une Étiquette pour une variable
Une étiquette pour une variable est l'identificateur du type de la variable. Elle permet de déterminer celui auquel elle appartient.
Une étiquette n'aura pas plus de 3 ou 4 caractères et sera représentée par un abrégé conventionnel comme celles présentes dans le tableau récapitulatif suivant :

Type	Préfixe	Exemple
Boolean	bln	**bln**IsEnabled
Byte	byt	**byt**Array
Collection	col	**col**Textboxes
Currency	cur	**cur**AmountDue
Date	dtm	**dtm**Birthday
Double	dbl	**dbl**Distance
Integer	int	**int**Quantity
Long	lng	**lng**Size
Object	obj	**obj**WordApplication
Single	sng	**sng**Average
String	str	**str**FullName
Type défini par l'utilisateur ou l'API	udt	**udt**BrowseInfo

Variant	vnt	**vnt**CharArray

Pour résumer...
Il est fortement recommandé de s'imposer ces règles. Pour ce qui est de la nomination des préfixes, il est très fréquent de lire du code avec des variables dont le préfixe est tronqué à 1 caractère au lieu de 3.
Sur le plan de la lisibilité, cela n'a pas d'importance si toutefois le développeur a respecté celle-ci dans tout son code.

Exemples:

Type	Préfixe	Exemple
Boolean	b	**b**IsEnabled
Double	d	**d**Distance
Integer	i	**i**Quantity
Long	l	**l**Size
Object	o	**o**WordApplication
String	s	**s**FullName
Type défini par l'utilisateur	u	**u**BrowseInfo
Variant	v	**v**CharArray

En revanche, cette nomination mono-caractère ne peut pas s'appliquer facilement pour les variables comme Byte, Collection, Currency, Date, Single... Donc il est préférable de se tenir la convention des 3 caractères.

Cas particulier de cohabitation variable mon-caractère et des variables dotées de préfixes mono-caractère et triple-caractères :
Les variables de type **objet** peuvent être déclarées avec la lettre **o**.
Les variables de type **Compteur** comme **I** n'ont pas de préfixe.

Sélectionnez

```
Dim oExcelApplication as Object
Dim oExcelWorkSheet as Object
Dim oRSTowns as DAO.Recordset
Dim strSQLGetTownList as String
Dim I as Integer
```

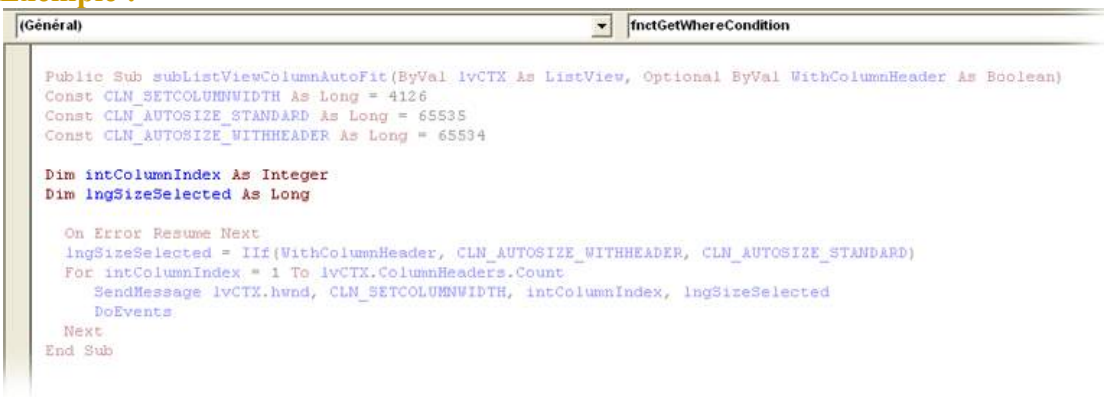
Exemple :

2-1-3. Exemples de déclaration de variables

Les variables déclarées dans une procédure ou une fonction

De manière générale, vous devez préfixer les variables déclarées au sein d'une fonction ou d'une procédure avec un préfixe qui en définit le type. Bien que très souvent ces dernières soient dépourvues de tout préfixe, je conseille de passer outre cette dérogation pour une meilleure lisibilité.

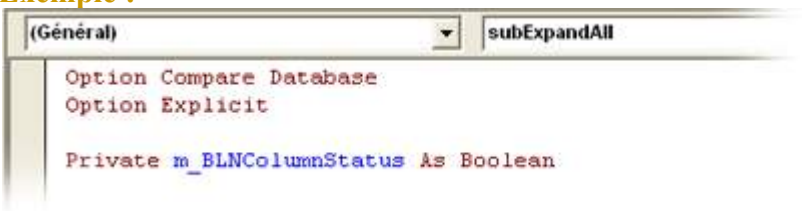
Exemple :



Les variables déclarées dans un module

Lorsque vous déclarez des variables de module, la convention universelle propose d'apposer un préfixe **m** suivi ou non d'un caractère de soulignement (Underscore) suivi du type de la variable et enfin, suivi d'un nom explicite.

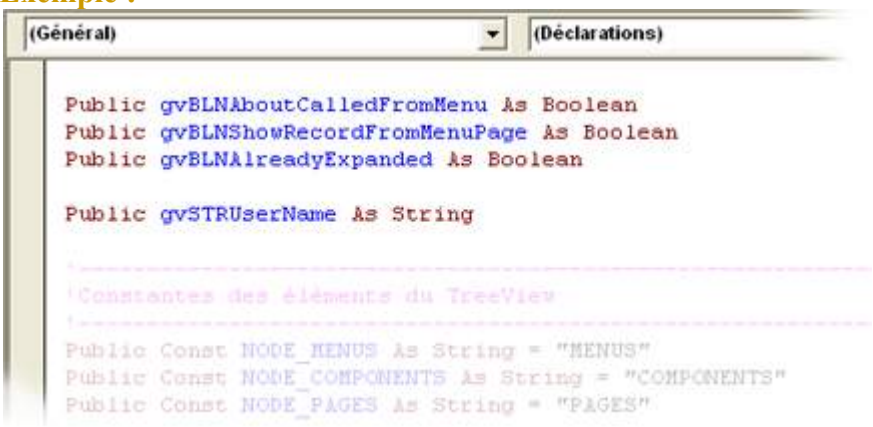
Exemple :



Les variables publiques

Lorsque vous déclarez des variables publiques dans un module elles sont exploitées dans l'ensemble du projet ; la convention universelle propose alors d'apposer un préfixe **g** suivi ou non d'un caractère de soulignement (Underscore) suivi du type de la variable et enfin, suivi d'un nom explicite. Il est envisageable d'adjoindre la lettre **v** derrière le **g** afin de préfixer **gv** devant le nom de la variable ce qui donne alors **gv**, sous entendu Global Variable.

Exemple :



Les variables de module de classe

Lorsque vous déclarez des variables dans une classe, la convention universelle propose d'apposer un préfixe **c** ou **cls** suivi ou non d'un caractère de soulignement (Underscore) suivi d'un nom explicite.

Exemple :

```
Option Compare Database
Option Explicit

Private m_ColumnStatus As Boolean

Private cNewInstance As clsApplication

Private Const LVM_FIRST As Long = &H1000
Private Const LVM_SETCOLUMNWIDTH As Long = (LVM_FIRST + 30)
```

2-1-4. Les constantes de module et les constantes en général

Lorsque vous déclarez des constantes dans un module, la convention universelle propose de les écrire en **MAJUSCULE** avec un nom explicite dont les mots qui constituent le nom peuvent être séparés par un caractère de soulignement (Underscore).

Elles peuvent être préfixées pour pouvoir identifier leur type ou leur appartenance commune à un groupe de fonctions par exemple.

Exemple :

```
Option Compare Database
Option Explicit

Private m_ColumnStatus As Boolean

Private Const LVM_FIRST As Long = &H1000
Private Const LVM_SETCOLUMNWIDTH As Long = (LVM_FIRST + 30)
Private Const LVSCW_AUTOSIZE As Long = -1
```

Les constantes pour les API's sont préfixées avec un terme connu des développeurs qui identifie leurs rôles :

Exemple pour l'Api exploitant le style d'une fenêtre :

Les constantes exploitant le style d'une fenêtre sont préfixées **SW_** signifiant **SHOW** :

Sélectionnez

```
Public Const SW_MAXIMIZE = 3
Public Const SW_MINIMIZE = 6
Public Const SW_NORMAL = 1
Public Const SW_HIDE = 0
```

Exemple pour l'Api exploitant la fonction MessageBox() (ou la fonction interne MsgBox d'ailleurs) :

Les constantes exploitant exploitant la fonction MessageBox() **MB_** signifiant **MsgBox** :

Sélectionnez

```
Public Const MB_ICONEXCLAMATION = &H30&
Public Const MB_ICONQUESTION = &H20&
Public Const MB_ICONINFORMATION = &H40&
Public Const MB_ICONSTOP = &H10&
Public Const MB_OK = &H0&
Public Const MB_OKCANCEL = &H1&
```

```
Public Const MB_YESNO = &H4&
```

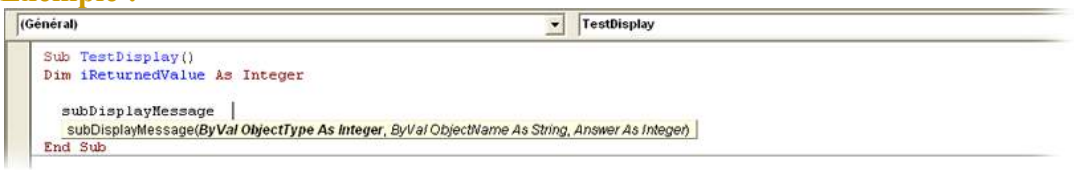
2-1-5. Les variables déclarées comme paramètre dans une procédure ou une fonction

Les variables déclarées comme paramètre dans une fonction ou dans une procédure ne sont en générale précédées d'aucun préfixe.

Exemple :

En effet, au moment de la saisie, le fait d'exploiter ou d'appeler une fonction ou une procédure provoque, par l'intermédiaire de l'**IntelliSense**, l'affichage du type des variables faisant office d'arguments.

Exemple :



3. Explications sur la portée d'une variable

Dans une application Visual Basic, les variables peuvent être de portée différente :

- **Locale** : Au sein d'une procédure ou d'une fonction

Sélectionnez

```
Dim MaVar As Type
```

- **Module** : En entête d'une feuille de module

Sélectionnez

```
Private m_MaVar As Type
```

- **Public** : Dans un module (si possible unique)

Sélectionnez

```
Public gvMaVar As Type
```

Pour ce qui est des variables publiques, je vous recommande de ne les utiliser que si vous n'avez pas d'autres possibilités de partager des valeurs entre les différents modules. Dans ce cas, il est conseillé de les regrouper ensemble au sein d'un même module (par exemple **basDeclaration**) et nommées avec un nom explicite.

Il est plus conventionnel (et plus propre) de mettre en place des propriétés (Property Let, Property Get).

Portée des déclarations

- 1 - Au sein **d'une procédure** ou d'une fonction les variables seront visibles uniquement dans la procédure ou la fonction dans laquelle elles sont déclarées.
- 2 - Au sein d'un module **les variables de module** (préfixées **m**) seront visibles uniquement dans la totalité du module dans lequel elles sont déclarées.
- 3 - Au sein d'un module **les variables publiques** (préfixées **g**) seront visibles dans la totalité de l'application.

3-1. Représentation des formats des types de données

Lorsque vous déclarez des variables, vous devez leur affecter un type.
Le tableau ci-dessous (inspiré de l'aide de Visual Basic) vous donne la liste des types et leur plage de valeur.

Toute variable déclarée sans spécification du type est déclarée en type **Variant.**

Type de données	Taille	Plage des valeurs
Byte	1 octets	0 à 255
Boolean	2 octets	True ou False
Integer	2 octets	-32 768 à 32 767
Long	4 octets	Nombre entier long : -2 147 483 648 à 2 147 483 647
Single	4 octets	Nombre à virgule flottante en simple précision : -3,452823E38 à -1,451298E-20 pour les valeurs négatives ; 1,451298E-20 à 3,452823E38 pour les valeurs positives;
Double	8 octets	Nombre à virgule flottante en double précision : -1,79769313486232E308 à -4,94565620841247E-324 pour les valeurs négatives ; 4,94565620841247E-324 à 1,79769313486232E308 pour les valeurs positives;
Currency	8 octets	Nombre entier à décalage : -922 337 203 685 477,5808 à 922 337 203 685 477,5807
Decimal	14 octets	+/-79 228 162 514 264 337 593 543 950 320 sans séparateur décimal ;
Date	8 octets	1er janvier 100 au 31 décembre 9999
Object	4 octets	Toute référence à des données de type **Object**

String	10 octets + longueur de la chaîne	0 à environ 2 milliards Pour les chaînes de longueur fixe : 1 à environ 65 450
Variant (numérique)	16 octets	Toute valeur numérique, avec la même plage de valeurs qu'une donnée de type **Double**
Variant (caractères)	22 octets + longueur de la chaîne	Même plage de valeurs qu'une donnée de type **String** de longueur variable
Type défini par l'utilisateur (Type)	En fonction des éléments	La plage de valeurs de chaque élément correspond à celle de son type de données.

4. Conventions générales concernant les noms

4-1. Les variables

Typiquement, les noms des variables sont au singulier. On applique le pluriel aux collections. Toutefois, il n'est pas obligatoire de respecter exclusivement cette règle. Pour ma part, j'utilise le **pluriel** pour des variables de type Long (Integer, Single, Double) devant me retourner par exemple une quantité et où le résultat attendu est au minimum égal à **1**.

Concernant la longueur du mot représentant le nom de la variable ou de l'objet, on tâchera de ne pas dépasser **15** caractères.
Exception concernant le nom des tables où la limite est fixée, toute dénomination incluse à **30** caractères et ce pour une compatibilité avec des bases de données externes comme SQL Server.

4-2. Les fonctions et procédures

Les procédures, les méthodes et les noms des fonctions sont écrits sous forme de **verbes.** Cela permet d'identifier le rôle de ces derniers.

Tout comme les objets, elles doivent être écrites avec une majuscule à chaque mot qui la constitue.
On pourra appliquer une terminologie scindée en **2** parties dans la cas ou l'on a besoin de dissocier les fonctions et procédures **publiques** des fonctions et procédures **privées** ou encore issues de **modules de classes**.

Exemple :

```
(Général)                                    GetCurrentUserName

Private Sub ChangeStatusButtons(ByVal Status As Boolean)
  cmdSave.Enabled = Not Status
  cmdNewRecord.Enabled = Status
  cmdShowAllRecord.Enabled = Not Status
  cmdSearchData.Enabled = Status
End Sub

Private Function GetCurrentUserName() As String
Const UNKNOWN_USER As String = "Utilisateur non défini"
Const MAX_LEN As Long = 255
Dim strBuffer As String
Dim lngReturn As Long

  strBuffer = Space(MAX_LEN)
  lngReturn = GetUserName(strBuffer, MAX_LEN)
  If lngReturn Then
      strBuffer = Left$(strBuffer, Len(strBuffer) - 1)
      strBuffer = StripNullChar(strBuffer)
  Else
      strBuffer = UNKNOWN_USER
  End If
  GetCurrentUserName = strBuffer
End Function

Function StripNullChar(ByVal Text As String) As String
Dim intPosition As Integer
  If InStr(1, Text, vbNullChar) > 0 Then
    intPosition = InStr(1, Text, vbNullChar) - 1
    StripNullChar = Trim(Left(Text, intPosition))
  Else
    StripNullChar = Text
  End If
End Function
```

- **ChangeStatusButtons** (Changer l'état des boutons)
- **GetCurrentUserName** (Obtenir le nom de l'utilisateur en cours)
- **StripNullChar** (Supprimer l'excédent de caractères de type NullChar)

Vous pouvez remarquer que je code systématiquement en anglais. C'est une habitude que je me suis donné parce qu'il n'est pas commode, selon moi, d'écrire le nom de variables, de fonctions et de procédures avec autant de clarté en langue francophone.
De plus, le problème de l'accentuation et des apostrophes se pose dans de nombreux cas. Il n'est évidemment pas conseillé **d'accentuer les voyelles** en programmation pour les nommage des objets en général.

- **ChangeStatusButtons** => (ChangerLEtatDesBoutons)
- **GetCurrentUserName** => (ObtenirLeNomDeLUtilsateurEnCours)
- **StripNullChar** => (SupprimerLesCaracteresNull)

On pourra enfin apposer un **Suffixe** déterminant une particularité de la fonction ou le la procédure. Ce dernier devient nécessaire dans le cas où des fonctions ou des procédures effectuent un traitement similaire mais attaquant un autre type de données par exemple.

Dans le langage **Visual Basic.Net**, on peut surcharger une procédure ou une fonction ce qui se résume à pouvoir employer **un même nom d'objet** pour chacune mais avec un nombre ou un type d'arguments différents.
Ceci n'est pas applicable en **VB** ni en **VBA** d'où la nécessité parfois, d'apposer un **Suffixe.**

5. Conventions générales concernant le nom des contrôles

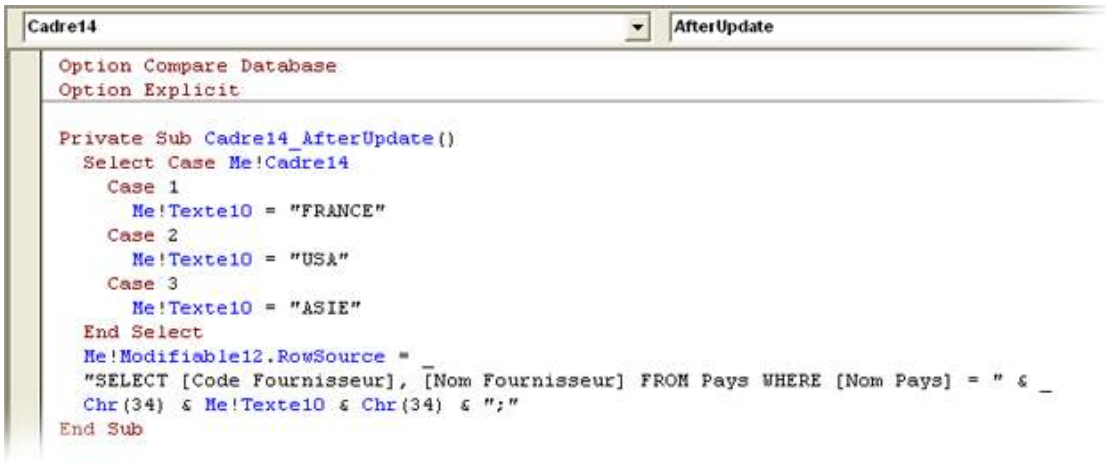
Tout comme les variables, les contrôles utilisés dans les formulaires (Forms) ou les états (Reports) doivent être nommés correctement.

Je vois trop souvent encore de nombreux développeurs qui ne prennent pas la peine de nommer les objets au sein des formulaires ou des états, voire les formulaires et les états eux-mêmes, ce qui provoque automatiquement la colère du relecteur qui se voit obligé de tenter de déchiffrer ce que l'auteur a écrit par le biais d'un basculement intempestif entre la feuille de code et le formulaire ou l'état.

J'ai même eu à reprendre, un jour, une base de données Access où des tables s'appelaient **Table** et ses champs s'appeler **Champ** et où aucun, mais absolument aucun contrôle ajouté n'était nommé correctement. Le projet de reprise aurait dû durer 5 jours : il a duré plus d'un mois.

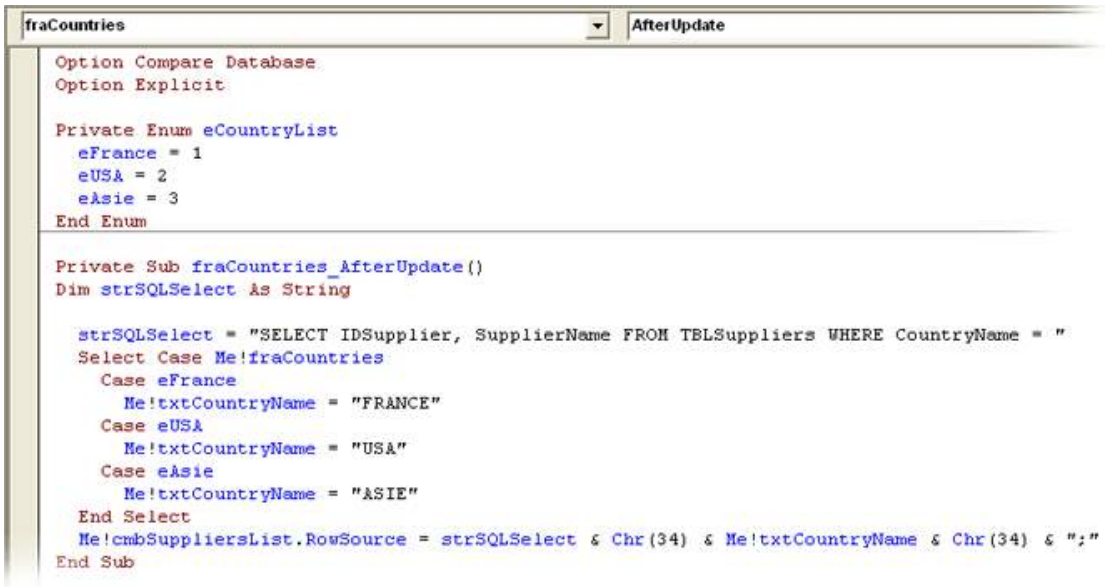
5-1. Exemple utilisant des objets non nommés :

Voyez par vous-même, bien que l'on comprenne ce bout de code, il est impossible ici de savoir quels contrôles sont sollicités.



5-2. Exemple utilisant des objets correctement nommés :

Ici, on comprend davantage le code et l'on sait tout de suite quel objet est sollicité ainsi que son type :



Vous pouvez remarquer également que le nom des champs et celui de la table sont en un seul mot (et ici en anglais) ce qui évite l'inclusion des crochets.
De toute façon, il n'est pas recommandé de nommer les objets avec des espaces.

5-3. Exemples d'appellation des contrôles avec leur préfixe

Un préfixe est à apposer devant le nom de l'objet selon le tableau suivant :

Exemple d'objet		Type d'objet	Préfixe

aniSendMail	=>	Animated button	ani
cboLanguages	=>	Combo box, drop-down list box	cbo
chkAlwaysOnTop	=>	Check box	chk
cmdSaveRecord	=>	Command button	cmd
comFax	=>	Communications	com
ctlAnyone	=>	Control (à utiliser lorsque le type de contrôle est non défini)	ctl
dbgCustomers	=>	Data-bound grid	dbg
dirDestination	=>	Directory list box	dir
dlgSelectFile	=>	Common dialog	dlg
drvList	=>	Drive list box	drv
filList	=>	File list box	fil
fraSubscription	=>	Frame	fra
frmPopupHistory	=>	Form	frm
grdProducts	=>	Grid	grd
hsbRate	=>	Horizontal scroll bar	hsb
ils32PixelsIcons	=>	ImageList	ils
imgProduct	=>	Image	img
keyNumlock	=>	Key status	key
lblFirstName	=>	Label	lbl
linSeparator	=>	Line	lin
lstCategories	=>	List box	lst
lvwComponents	=>	ListView	lvw
mciCDPlay	=>	MCI	mci
mdiDocument	=>	MDI child form	mdi
mnuEdit	=>	Menu	mnu
mpmSendMessage	=>	MAPI message	mpm
msgSheetGrid	=>	MS Flex grid	msg
mstChangeSettings	=>	MS Tab	mst
oleWorkbook	=>	OLE	ole
picCompanyLogo	=>	Picture	pic
pnlBottomButtons	=>	3D Panel	pnl
prgLoadingRecords	=>	ProgressBar	prg
rptSalesByMonth	=>	Report	rpt
shpRectangle	=>	Shape	shp
sldSoundVolume	=>	Slider	sld

spnMonths	=>	Spin	spn
staInformations	=>	StatusBar	sta
tabSetPreferences	=>	TabStrip	tab
tlbStandard	=>	Toolbar	tlb
tmrUpdateRecord	=>	Timer	tmr
trwFolders	=>	TreeView	trw
txtCompany	=>	Text box	txt
vsbRate	=>	Vertical scroll bar	vsb

Cette normalisation est régie par des standards issus de conventions préétablies.
Bien que Microsoft Access (entre autres...) accepte toutes les fantaisies en matière de nommage des objets qui composent une application, il est recommandé de perdre cette facilité pour s'adonner à de bonnes habitudes.

Pour information, notez bien que ce même Microsoft Access propose, à ma grande déception, des noms de **contrôles accentués** autant que des **noms de champs** dotés d'espaces dans ses bases de données exemples nommées **Comptoir.mdb** et **Northwind.mdb** ce qui est un très mauvais pas pour les débutants qui ne connaissent pas le produit et suivent tout bêtement ce qui est indiqué...

Par voie de conséquence et bien que cela n'ait pas d'incidence majeure au sein de l'application Access elle-même, ce n'est pas une référence et c'est une erreur presque impardonnable de la part de Microsoft d'autant plus que la propriété Légende permet d'étiqueter ses champs comme bon nous semble et ainsi, éviter par exemple, de nommer les champs avec des espaces.

6. Les autres conventions

Il existe un grand nombre de conventions possibles pour les objets en général. Au vu de la multitude des objets présents dans un outil de développement, il est préférable de se documenter dans l'aide pour obtenir des détails. A titre d'exemple, je liste ci-dessous les conventions à appliquer pour les objets utilisés pour **DAO.**

Exemple d'objet		**Type d'objet**	**Préfixe**
conAllForms	=>	Container	con
dbRentCars	=>	Database	db
dbeMSJet	=>	DBEngine	dbe
docBill	=>	Document	doc
fldTown	=>	Field	fld
idxPostalCode	=>	Index	idx
prmIDKey	=>	Parameter	prm
qrySalesByCustomers	=>	QueryDef	qry
rstSuppliers	=>	Recordset	rst
relRentCars	=>	Relation	rel
tdfNewTable	=>	TableDef	tdf
wksDBAccess	=>	Workspace	wks

7. Les conventions de nommage des champs

Je n'aborde ce sujet qu'en dernier bien qu'il soit un des plus importants au sein d'une base de données. Il est effectivement impératif de respecter les normes en matière de définition de nom des champs d'une table.

Les 6 règles à retenir pour nommer un champ

1. Ne jamais mettre d'espaces dans un nom de champ ;
2. Ne jamais mettre de caractères spéciaux dans un nom de champ (N° - & - % - @ etc...) ;
3. Ne jamais accentuer les voyelles dans un nom de champ ;
4. Faire en sorte qu'une certaine homogénéité au niveau de la longueur soit constante (Les champs **DBase** font 13 caractères) ;
5. Ne pas abuser des underscores (_) pour séparer les termes ;
6. Être aussi explicite que possible dans le nom du champ ;

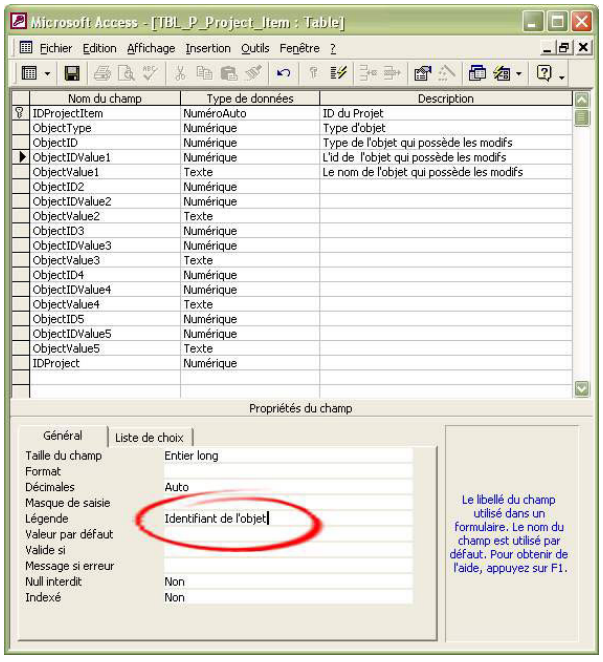
De manière générale, je nomme aussi mes noms de champs en Anglais. Cela permet de respecter une partie des règles énumérées ci-dessus.

Un exemple concret à ne surtout pas suivre est bien entendu la base de données exemple nommées "**les Comptoirs**" qui est livrée avec Access.

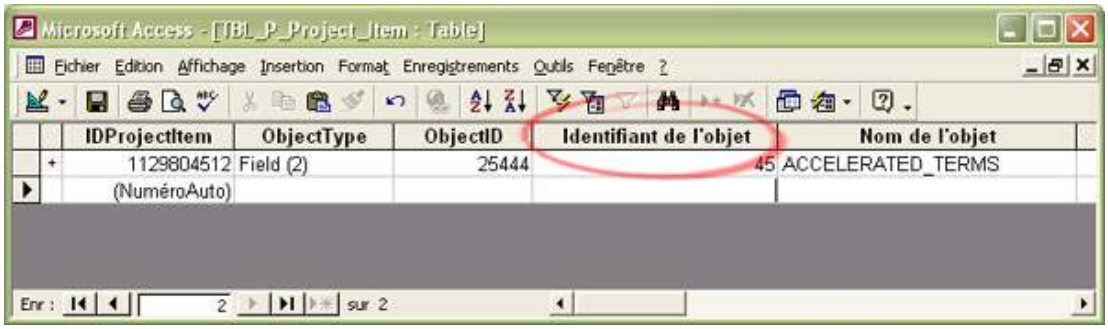
7-1. Comment faire en sorte que le nom affiché ne soit pas celui du champ ?

Effectivement, en s'imposant ces règles, le nom du champ apparaît par défaut dans vos formulaires et vos états ce qui, je vous l'accorde n'est pas joli d'une part et n'est pas explicite d'autre part.
Par exemple un nom de champ nommé "**BirthDate**" fera apparaître par défaut "**BirthDate**" dans un formulaire ou un état.

Si l'on veut voir mentionner "**Date de naissance**" il faut alors alimenter la propriété **Légende** dans la table comme le montre la figure ci-dessous:



En mode feuille de données, vous constatez tout de suite que le nom de la colonne s'approprie la légende et non le nom du champ.



L'énorme avantage d'user de cette propriété est que vous pouvez cette fois user de toutes les fantaisies pour renseigner cette propriété **Légende**:

Les 5 règles à retenir pour la propriété légende

1. Vous pouvez mettre des espaces dans une légende ;
2. Vous pouvez mettre de caractères spéciaux dans une légende (N° - & - % - @ etc...) ;
3. Vous pouvez accentuer les voyelles dans une légende ;
4. Vous pouvez être aussi explicite que possible dans une légende ;
5. Vous pouvez faire en sorte de respecter une certaine homogénéité au niveau de la longueur des légendes (Ce n'est pas une phrase !) ;

8. Quelques recommandations complémentaires

Pour une convivialité de lecture, je vous recommande d'écrire les mots en entier pour être plus explicite.

Par exemple :

Sélectionnez

```
Dim strCustomerName As String
```

Au lieu de

Sélectionnez

```
Dim strCustName As String
```

En revanche, il se peut que la longueur d'un mot désignant une variable soit trop long auquel cas, pour les mots connus, vous pouvez abréger :

Par exemple :

Sélectionnez

```
Dim lngProductID As Long
```

Au lieu de

Sélectionnez

```
Dim lngProductCodeIdentifier As Long
```

Afin de séparer les mots désignant une variable, il est conseillé de ne pas utiliser le caractère de soulignement (Underscore) mais plutôt de mettre une **MAJUSCULE** à chaque mot.

Par exemple :

Sélectionnez

```
Dim strUserLastName As Long
```

Au lieu de

Sélectionnez

```
Dim str_user_last_name As Long
```

On notera qu'il est toutefois possible de mettre un caractère de soulignement pour le suffixe.

Par exemple :

Sélectionnez

```
Private Function GetFullCustomerList_Sorted(ByVal Order As String, Optional ByVal
TopCount As Integer = 0) As String
Dim strSQLCustomers As String
Dim strSQLOrderBy As String

  If TopCount Then
    strSQLCustomers = "SELECT TOP " & TopCount & " Gender, FirstName, LastName FROM
TBLCustomers"
  Else
    strSQLCustomers = "SELECT Gender, FirstName, LastName FROM TBLCustomers"
  End If
  Select Case Order
    Case SQL_ORDER_AZ
      strSQLOrderBy = "ORDER BY LastName ASC"
    Case SQL_ORDER_ZA
      strSQLOrderBy = "ORDER BY LastName DESC"
  End Select
  strSQLCustomers = strSQLCustomer & vbCrLf & strSQLOrderBy
  GetFullCustomerList = strSQLCustomers
End Function
```

8-1. Les variables de type Entier par défaut (Integer)

Il est très fréquent lire des blocs de code avec des variables **I, J** ou **N** déclarées en entier (Integer). C'est une convention typique comme en mathématiques, les professeurs définissaient les points d'une droite x et y.

8-2. Les instructions Deftype

Ce type d'instruction est utilisé au **niveau du module** et définit le type de donnée par défaut pour l'ensemble des variables, des arguments passés aux procédures et le type de renvoi pour les procédures Function et Property Get dont le nom commence avec les caractères spécifiés.
Ceci permet dans un sens de ne pas avoir à déclarer des variables mais ce type de déclaration implicite est à proscrire car il est peut parlant et difficile à interpréter.
De plus il oblige à chaque fois à se souvenir que Telle Variable, Telle Procédure [...] qui commence par Tel Caractère est de Tel Type, d'où une conception et une approche du code délicate.

Exemple :

8-2-1. Le nom de l'instruction détermine le type de données

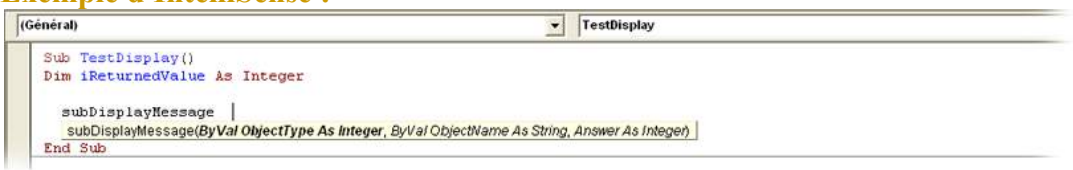
Instruction Def	Type de données
DefBool	Boolean
DefByte	Byte
DefInt	Integer
DefLng	Long
DefCur	Currency
DefSng	Single
DefDbl	Double
DefDate	Date

8-3. L'utilisation de l'IntelliSense

Usez et abusez de l'**IntelliSense**. Cet outil magique est traduit par l'auto-complémentation de la syntaxe (auto-completion) au moment de la saisie du code.

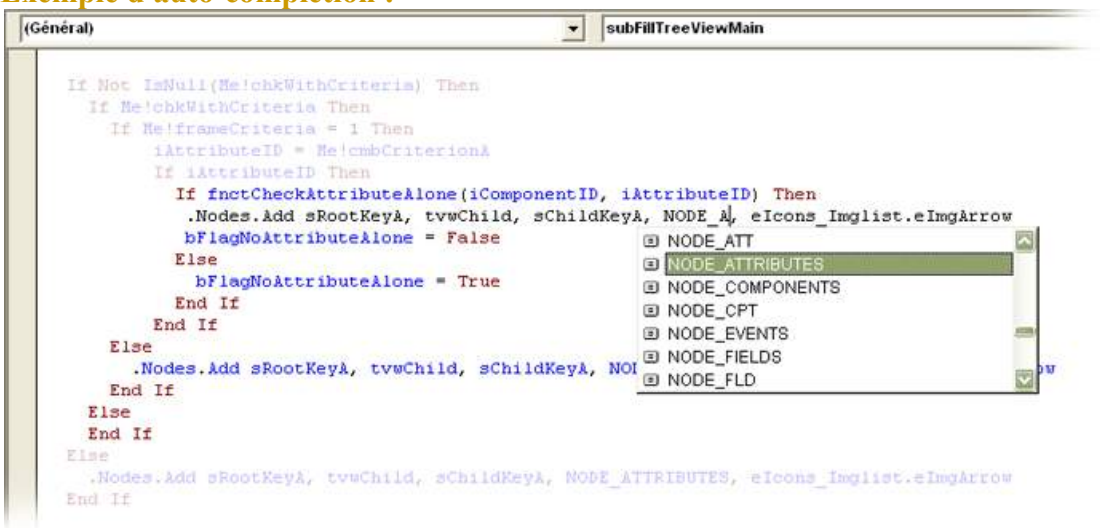
- Si la saisie en cours est une procédure ou une fonction déjà présente, l'**IntelliSense** vous affiche une info bulle de couleur jaune pour vous indiquer les arguments faisant office de paramètre à lui passer.

Exemple d'IntelliSense :



- Si la saisie en cours est une variable, une constante ou encore une énumération, [...], l'**Auto-Completion** vous affiche une liste des objets correspondants.

Exemple d'auto-completion :



Pour activer l'Auto-Completion, au moment de l'écriture des premiers caractères de l'objet, appuyez sur les touches **Ctrl+Espace** simultanément.

9. Conclusion

Ce document n'est qu'un bref récapitulatif pour les conventions de nommage.
Il s'avère toutefois utile d'en prendre note avec un certain sérieux dans le sens où vous serez bien heureux de pouvoir faire évoluer vos projets.
La reprise et l'évolution des ces derniers ne seront que facilitées par la simple mise en application de ces petites règles qui, en finalité, deviennent très vite une (bonne) habitude.