LibreOffice
The Document Foundation

Guide Calc

Chapitre 12
Macros Calc

Automatiser les tâches répétitives

La documentation pour LibreOffice est disponible à fr.libreoffice.org

Copyright

Contributeurs

Andrew Pitonyak Barbara Duprey Hal Parker Simon Brydon

Traducteur

Christian Chenal

Relecteurs : Philippe Clément, Pierre-Yves Samyn

Retours

Veuillez envoyer vos commentaires ou suggestions à propos de ce document à :
discuss@fr.libreoffice.org

Remerciements

Ce chapitre est basé sur le Chapitre 12 de *OpenOffice.org 3.3 Calc Guide*. Les contributeurs à ce chapitre sont :

Andrew Pitonyak Gary Schnabl
Jean Hollis Weber Claire Wood

Date de publication et version du logiciel

Publié le 11 juillet 2012. Basé sur LibreOffice 3.5.4.

Note pour les utilisateurs Mac

Certaines combinaisons de touches et certains éléments de menus sont différents entre un Mac et Windows ou Linux. La table ci-dessous donne quelques équivalents pour les instructions de ce chapitre. Pour une liste plus détaillée, voyez dans l'Aide.

Windows/Linux	***Équivalent Mac***	***Effet***
Sélection de menu **Outils > Options**	**LibreOffice > Préférences**	Accès aux options de paramétrage
Clic droit	⌘+*clic*	Ouvre un menu contextuel
Ctrl (Contrôle)	⌘ *(Commande)*	Utilisé avec d'autres touches
F5	*Maj*+⌘+*F5*	Ouvre le Navigateur
F11	⌘+*T*	Ouvre la fenêtre Styles & Formatage

Table des matières

Introduction

Une macro est une séquence enregistrée de commandes ou de saisies au clavier qui sont conservées pour une utilisation ultérieure. Un exemple de macro simple serait une macro qui "saisit" votre adresse. Le langage de macro LibreOffice est très souple et permet l'automatisation de tâches simples ou complexes. Les macros sont particulièrement utiles pour répéter une suite d'actions exactement de la même façon, plusieurs fois au cours du temps.

Les macros sont regroupées en *modules* et *bibliothèques.*

- Un module contient un ensemble de macros.
- Une bibliothèque est un regroupement de modules, cette organisation permettant de mieux ranger les macros contenues dans les modules.

LibreOffice crée par défaut une bibliothèque nommée Standard dans chaque document. Si vous ne créez que peu de macros, vous pourrez n'utiliser que cette bibliothèque et un seul module.

Ce chapitre va aborder brièvement les problèmes habituels relatifs à la programmation de macros dans Calc.

Utiliser l'enregistreur de macro

Le Chapitre 13 (Débuter avec les Macros) du *Guide du Débutant* apporte les connaissances de base pour comprendre les possibilités générales des macros de LibreOffice en utilisant l'enregistreur de macro. L'exemple qui va suivre comportera des explications moins détaillées. Les étapes suivantes créent une macro qui effectue un collage spécial avec une multiplication.

Attention 	Pour pouvoir utiliser l'enregistreur de macro, vous devez tout d'abord activer l'option *Activer les fonctions expérimentales (non stabilisées)* qui se trouve dans **Outils > Options > LibreOffice > Général**.

1) Ouvrez un nouveau classeur.
2) Saisissez des nombres dans une feuille.

	A	B	C
1	1	8	9
2	2	7	10
3	3	6	11
4			

Figure 1 : Saisie des nombres

3) Sélectionnez la cellule A3, qui contient le nombre 3, et copiez cette valeur dans le presse-papiers.
4) Sélectionnez la plage A1:C3.
5) Utilisez **Outils > Macros > Enregistrer une macro** pour démarrer l'enregistreur de macro. La boîte de dialogue Enregistrer une macro s'affiche avec un bouton **Terminer l'enregistrement**.

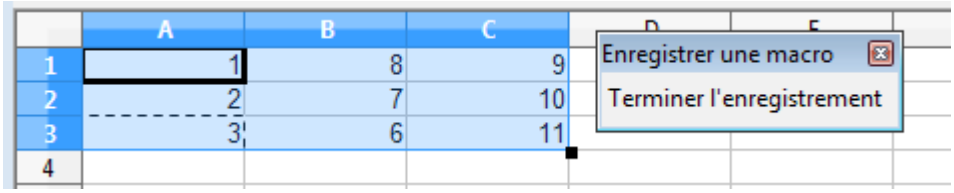


Figure 2 : Bouton Terminer l'enregistrement

6) Utilisez **Édition > Collage spécial** pour ouvrir la boîte de dialogue Collage spécial (Figure 3).

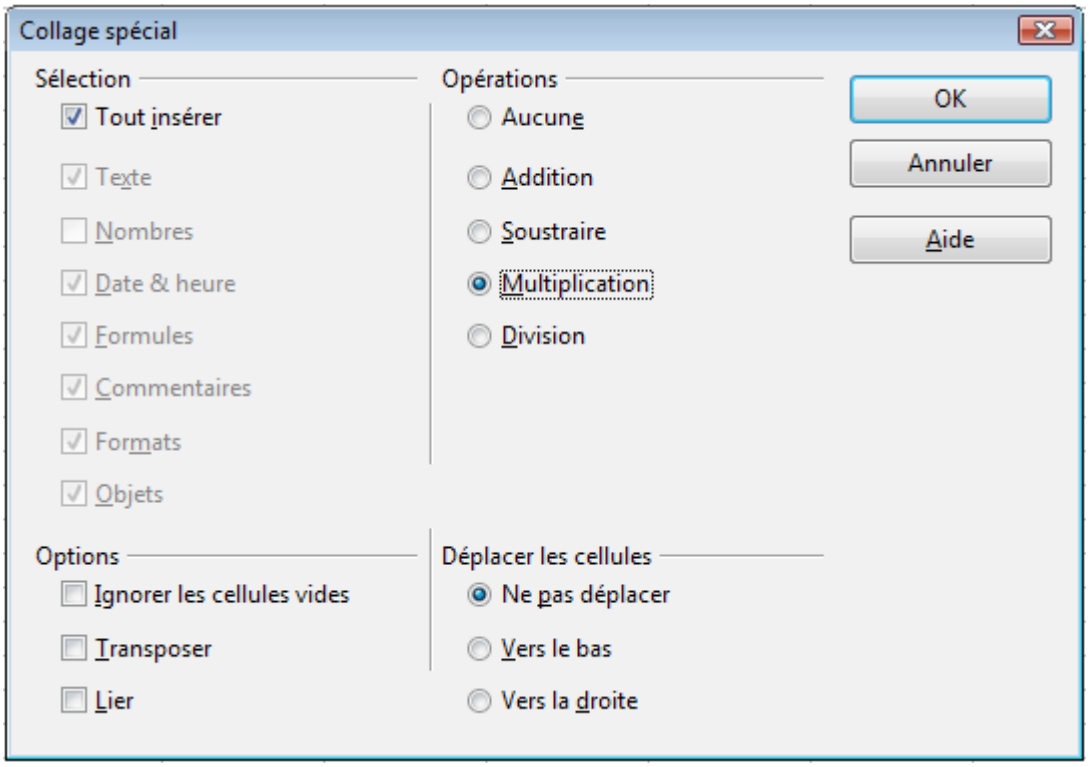


Figure 3 : Boîte de dialogue Collage spécial

7) Choisissez **Multiplier** dans la section *Opérations* et cliquez sur **OK**. Les cellules sont alors multipliées par 3 (Figure 4).

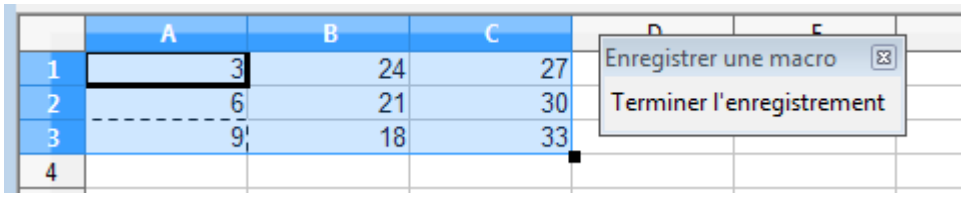


Figure 4 : Cellules multipliées par 3

8) Cliquez sur **Terminer l'enregistrement** pour arrêter l'enregistreur de macro. La boîte de dialogue Macros LibreOffice Basic s'ouvre (Figure 5).
9) Sélectionnez le document en cours (voir Figure 5). Pour cet exemple, le document Calc en cours s'intitule *Sans nom 1*. Les documents comportent une bibliothèque appelée Standard. Cette bibliothèque n'est pas affichée tant que ce n'est pas nécessaire pour Calc, et donc à cet instant votre nouveau document ne montre pas de bibliothèque. Vous pouvez en créer une nouvelle, mais ce n'est pas indispensable.

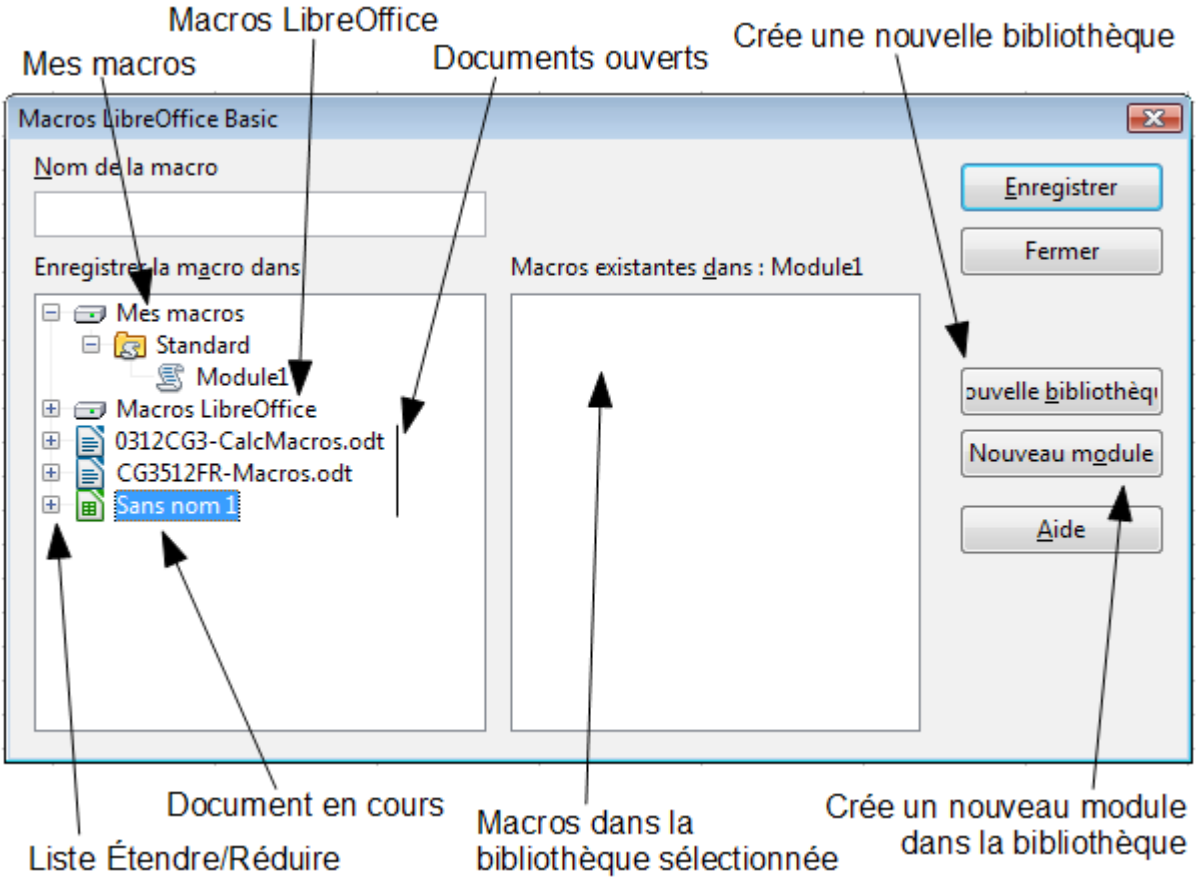
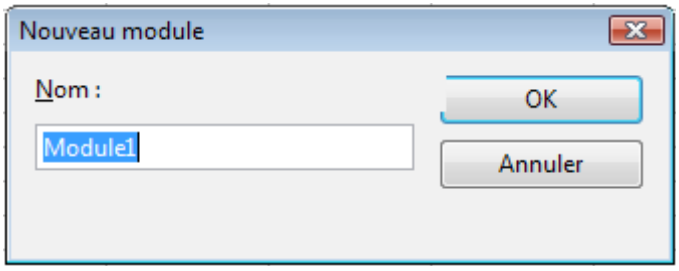


Figure 5 : Boîte de dialogue Macros LibreOffice Basic

10) Cliquez sur **Nouveau module**. Si aucune bibliothèque n'existe, la bibliothèque Standard est automatiquement affichée et utilisée. Dans la boîte de dialogue Nouveau module, saisissez un nom pour le nouveau module ou laissez le nom par défaut.



Note	Les noms de bibliothèques, modules, macros doivent respecter des règles strictes. Pour se limiter aux principales, les noms doivent : • commencer par une lettre, • ne pas contenir d'espace, • ne pas contenir de caractères spéciaux, accents compris, hormis le _ (tiret bas).

11) Cliquez sur **OK** pour créer un nouveau module appelé Module1. Sélectionnez le Module1 nouvellement créé, saisissez **`CopieMultiplication`** dans la zone *Nom de la macro* en haut à gauche, et cliquez sur **Enregistrer** (Figure 6).

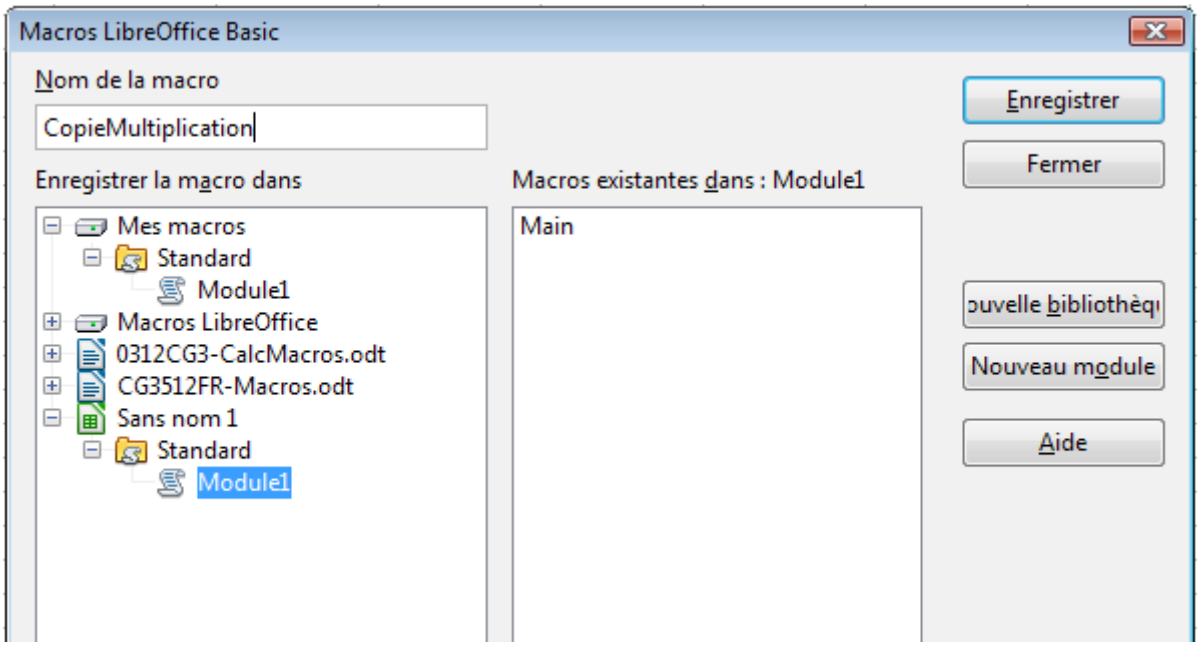


Figure 6 : Sélectionnez le module et nommez la macro

La macro créée est enregistrée dans Module1 de la bibliothèque Standard du document *Sans nom 1*. Le Listing 1 vous montre le contenu de la macro.

Listing 1. Collage spécial avec multiplication

```
sub CopieMultiplication
rem ----------------------------------------------------------------------
rem define variables
dim document   as object
dim dispatcher as object
rem ----------------------------------------------------------------------
rem get access to the document
document   = ThisComponent.CurrentController.Frame
dispatcher = createUnoService("com.sun.star.frame.DispatchHelper")

rem ----------------------------------------------------------------------
dim args1(5) as new com.sun.star.beans.PropertyValue
args1(0).Name = "Flags"
args1(0).Value = "A"
args1(1).Name = "FormulaCommand"
args1(1).Value = 3
args1(2).Name = "SkipEmptyCells"
args1(2).Value = false
args1(3).Name = "Transpose"
args1(3).Value = false
args1(4).Name = "AsLink"
args1(4).Value = false
args1(5).Name = "MoveMode"
args1(5).Value = 4

dispatcher.executeDispatch(document, ".uno:InsertContents", "", 0,
args1())

end sub
```

Vous trouverez plus de détails sur l'enregistrement des macros dans le Chapitre 13 (Débuter avec les Macros) du *Guide du Débutant*. Nous vous recommandons de le lire si vous ne l'avez déjà fait. Vous trouverez également plus de détails dans les paragraphes suivants, mais nous ne reviendrons plus sur l'enregistrement des macros.

Écrire vos propres fonctions

Calc peut appeler des macros en tant que fonctions Calc. Suivez les étapes suivantes pour créer une macro simple :

1) Créez un nouveau document Calc appelé **`CalcTestMacros.ods`**.
2) Utilisez **Outils > Macros > Gérer les macros > LibreOffice Basic** pour ouvrir la boîte de dialogue Macros LibreOffice Basic. La zone *Macro de* liste les conteneurs de bibliothèques de macros disponibles, dont les documents LibreOffice ouverts. *Mes macros* contient les macros que vous avez écrites ou ajoutées à LibreOffice. *Macros LibreOffice* contient les macros fournies avec LibreOffice, qui ne peuvent être que consultées.

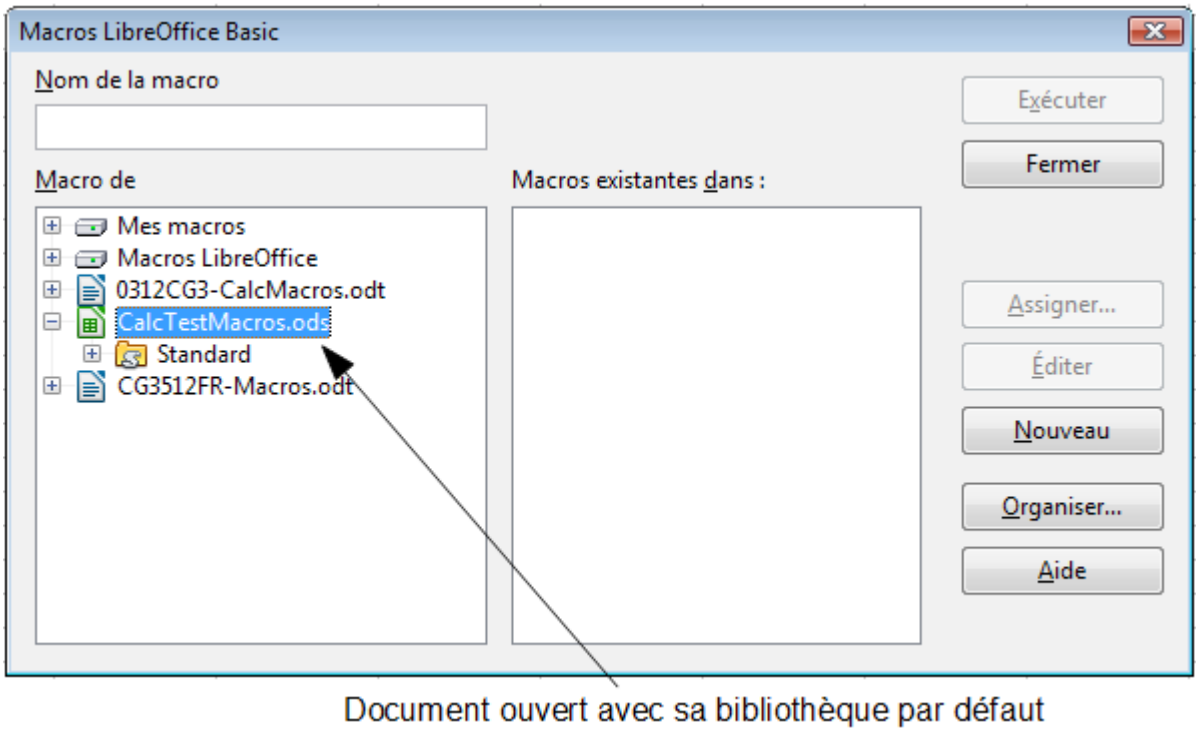


Figure 7 : Boîte de dialogue Macros LibreOffice Basic

3) Cliquez sur **Organiser** pour ouvrir la boîte de dialogue Gestion des macros de LibreOffice Basic (Figure 8). Dans l'onglet *Bibliothèques*, sélectionnez le document qui va contenir la macro.

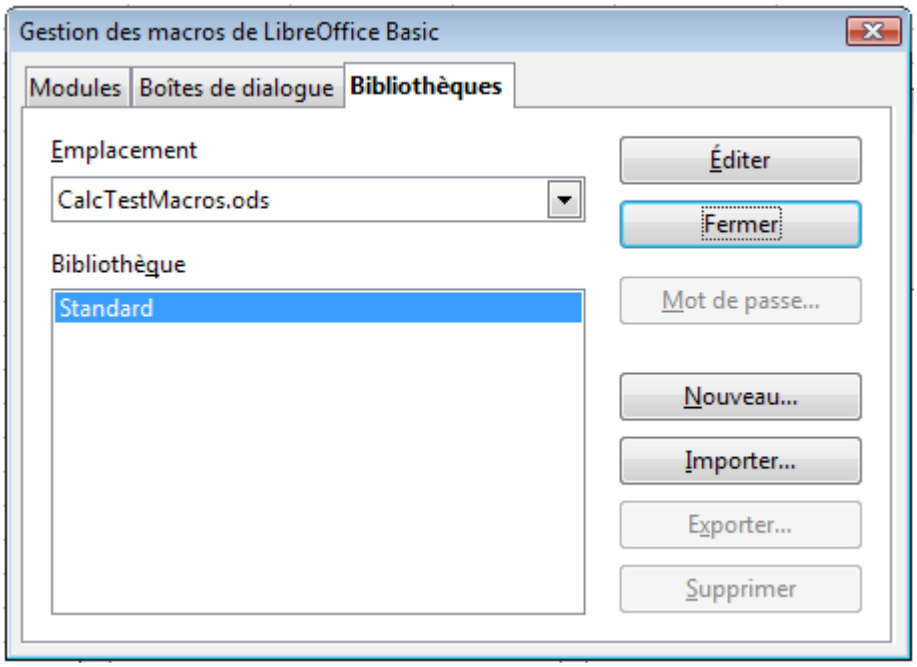


Figure 8 : Gestion des macros de LibreOffice Basic

4) Cliquez sur Nouveau pour ouvrir la boîte de dialogue Nouvelle bibliothèque.

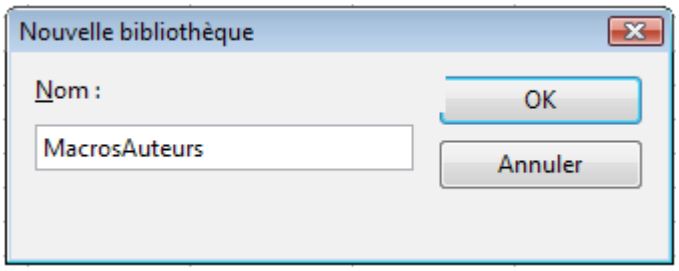


Figure 9 : Boîte de dialogue Nouvelle bibliothèque

5) Saisissez un nom de bibliothèque significatif (comme MacrosAuteurs) et cliquez sur OK pour créer la bibliothèque. Le nouveau nom de bibliothèque est affiché dans la liste des bibliothèques, mais la boîte de dialogue peut n'afficher qu'une partie du nom.

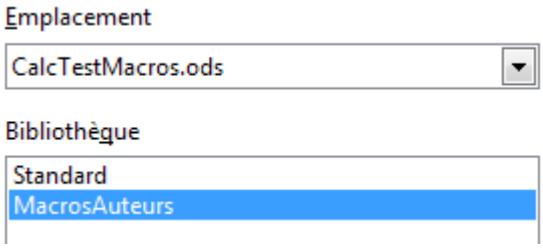


Figure 10 : La bibliothèque est affichée

6) Sélectionnez MacrosAuteurs et cliquez sur **Éditer** pour éditer la bibliothèque. Calc crée automatiquement un module appelé Module1 et une macro appelée Main.

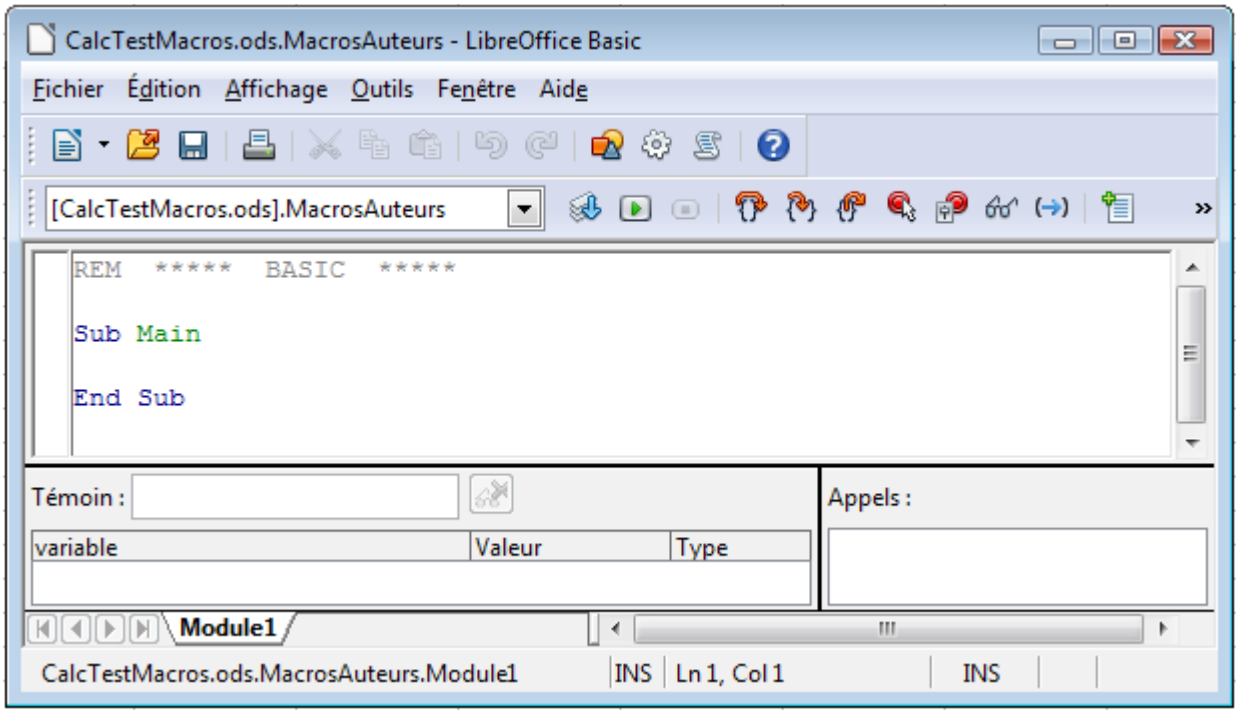


Figure 11 : Environnement de Développement Intégré Basic

7) Modifiez le code pour qu'il soit celui du Listing 2. Vous écrivez la fonction **`NombreCinq`**, qui retourne le nombre 5. L'instruction `Option Explicit` oblige à ce que toutes les variables soient déclarées avant leur utilisation. Si cette instruction est omise et les variables non déclarées, les variables seront automatiquement définies avec le type Variant pour leur première utilisation.

Note	Une *variable* est un emplacement que le programmeur réserve en mémoire afin d'y "déposer" des valeurs durant l'exécution de la macro.

Attention	L'instruction `Option Explicit` doit être écrite au début du module, **avant toute instruction**. Elle ne peut être précédée que de commentaires (ligne commençant une apostrophe ou par REM), comme dans l'exemple du Listing 2.

8) Enregistrez le Module1 ainsi modifié.

Listing 2. Fonction qui retourne le nombre 5

```
REM  *****  BASIC  *****
Option Explicit

Sub Main

End Sub

Function NombreCinq()
    NombreCinq = 5
End Function
```

Utiliser une macro en tant que fonction

Dans le document CalcTestMacros.ods, saisissez la formule =NombreCinq() (Figure 12). Calc retrouve la macro et l'appelle.

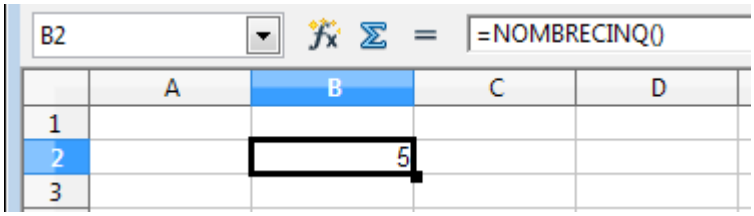


Figure 12 : Utilisation de la macro NombreCinq() en tant que fonction Calc

Note	Les noms de fonctions ne sont pas sensibles à la casse. Dans la Figure 12, vous pouvez saisir `=NombreCinq()` et Calc affichera `=NOMBRECINQ()`.

Enregistrez le document Calc, fermez le et ouvrez le à nouveau. En fonctions de vos paramètres dans **Outils > Options > LibreOffice > Sécurité** section *Sécurité des macros*, Calc va afficher le message d'avertissement de la Figure 13 ou celui de la Figure 14. Vous devrez cliquer sur **Activer les macros**, ou Calc ne permettra pas qu'une macro soit exécutée à l'intérieur du document. Si vous ne vous attendiez pas à ce que le document contienne une macro, il est plus sûr de cliquer sur **Désactiver les macros**, pour le cas où la macro contiendrait du code malveillant.

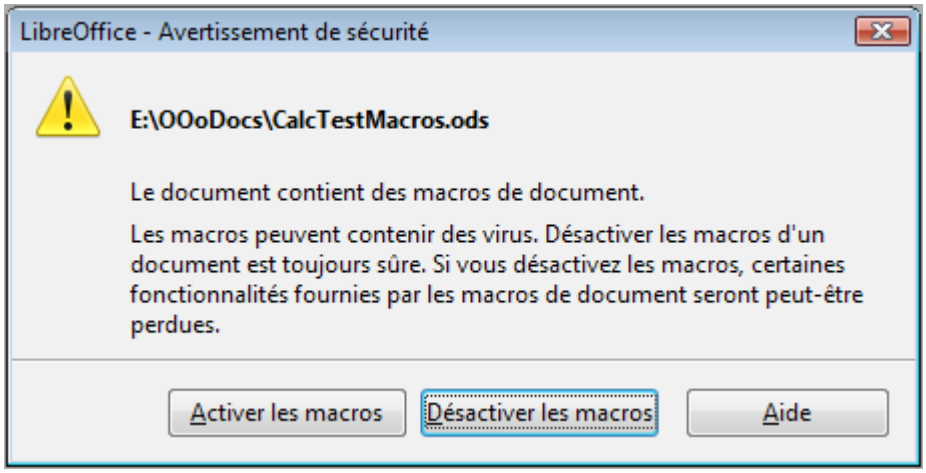


Figure 13 : Avertissement que le document contient des macros

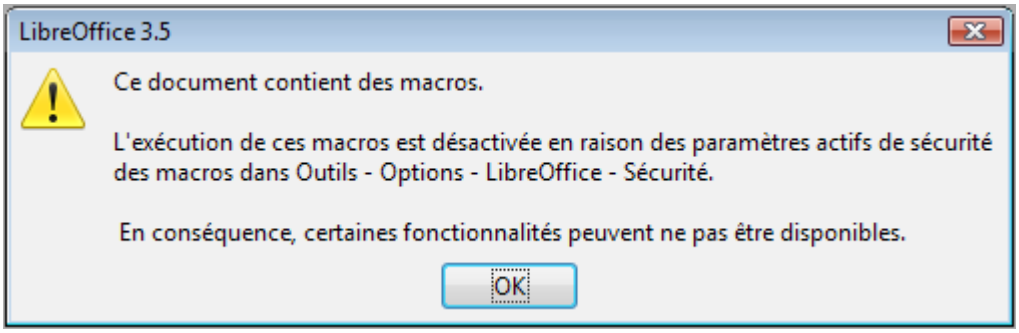


Figure 14 : Avertissement que des macros ont été désactivées

Si vous choisissez de désactiver les macros, alors, quand le document se charge, Calc ne peut plus trouver la fonction.

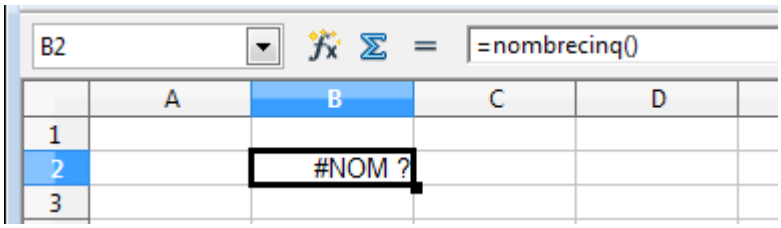


Figure 15 : La fonction n'est pas disponible

Quand un document est créé et enregistré, il contient automatiquement une bibliothèque appelée Standard. Cette bibliothèque Standard est automatiquement chargée quand le document est ouvert. Aucune autre bibliothèque n'est chargée automatiquement.

Calc ne comporte pas de fonction appelée NombreCinq(), et il va donc rechercher cette fonction dans toutes les bibliothèques de macros ouvertes et visibles. Calc va explorer les bibliothèques dans *Macros LibreOffice*, *Mes macros* et la bibliothèque Standard du document Calc (voir Figure 7). La fonction NombreCinq() est stockée dans la bibliothèque MacrosAuteurs, qui n'est pas chargée automatiquement à l'ouverture du document, ce qui explique que la fonction n'est pas exécutée.

Utilisez **Outils > Macros > Gérer les macros > LibreOffice Basic** pour ouvrir la boîte de dialogue Macros LibreOffice Basic (Figure 16). Développez CalcTestMacros et trouvez MacrosAuteurs. Notez que les icônes sont de couleurs différentes (jaunes ou grisées) selon que la bibliothèque soit chargée ou non.

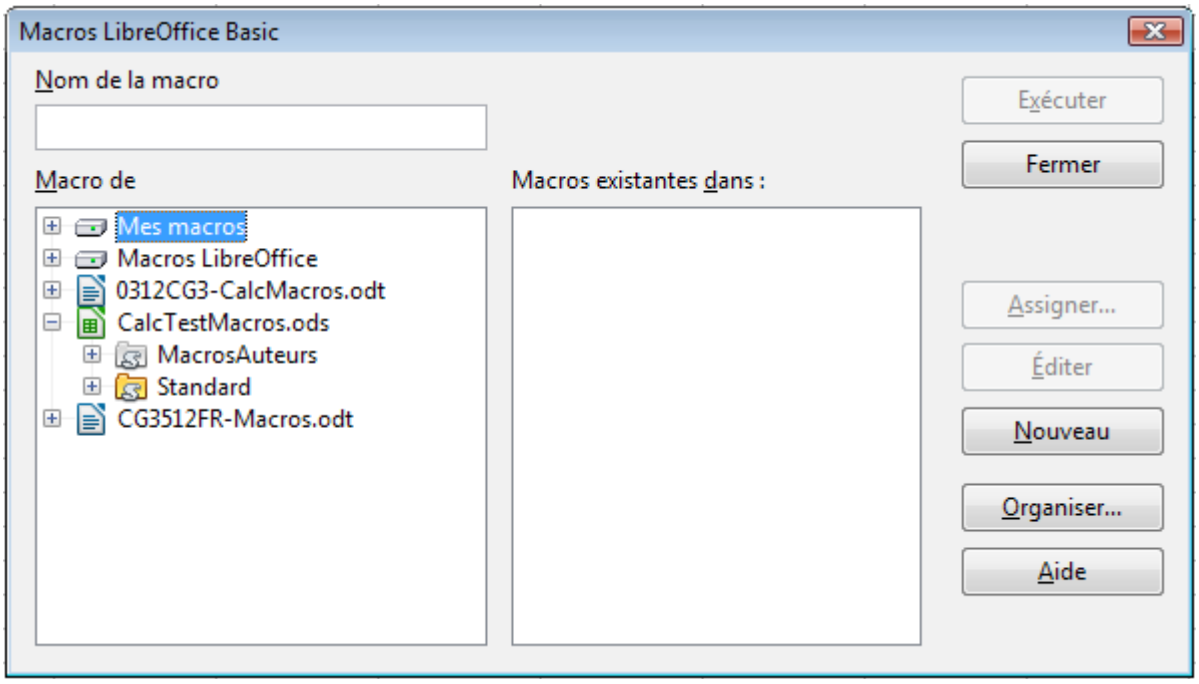


Figure 16 : Couleurs des bibliothèques chargées ou non

Cliquez sur le symbole d'extension (habituellement le signe + ou un triangle ▶) devant MacrosAuteurs pour charger la bibliothèque. La couleur de l'icône change pour indiquer que la bibliothèque est désormais chargée. Cliquez sur **Fermer** pour fermer la boîte de dialogue.

Malheureusement, les cellules qui contiennent `=NombreCinq()` sont toujours en erreur. Calc ne recalcule pas les cellules en erreur, à moins que vous ne les modifiiez ou que vous les changiez d'une façon ou d'une autre. La meilleure solution est de stocker les macros utilisées en tant que fonctions dans la bibliothèque Standard. Si la macro a une taille importante ou s'il y a beaucoup de

macros, un relais avec le nom voulu peut être stocké dans la bibliothèque Standard. La macro relais charge la bibliothèque contenant l'implémentation, puis appelle cette implémentation.

1) Utilisez **Outils > Macros > Gérer les macros > LibreOffice Basic** pour ouvrir la boîte de dialogue Macros LibreOffice Basic. Sélectionnez la macro NombreCinq et cliquez sur **Éditer** pour ouvrir la macro en édition.
2) Modifiez le nom NombreCinq en NombreCinq_Implementation (Listing 3).

Listing 3. Nom changé de NombreCinq en NombreCinq_Implementation

```
Function NombreCinq_Implementation() as integer
    NombreCinq_Implementation = 5
End Function
```

3) Dans l'EDI Basic, cliquez sur le bouton **Sélectionner une macro**  pour ouvrir la boîte de dialogue Macros LibreOffice Basic.
4) Sélectionnez la bibliothèque Standard du document CalcTestMacros et cliquez sur **Nouveau** pour créer un nouveau module. Saisissez un nom significatif comme FonctionsCalc et cliquez sur **OK**. LibreOffice crée automatiquement une macro appelée Main et ouvre le module pour édition.
5) Créez une macro dans la bibliothèque Standard qui appelle l'implémentation de la fonction (voir Listing 4). La nouvelle macro charge la bibliothèque MacrosAuteurs si ce n'est pas déjà fait, et ensuite appelle l'implémentation de la fonction.

Listing 4. Macro NombreCinq de la bibliothèque Standard

```
Function NombreCinq()
    If Not BasicLibraries.isLibraryLoaded("MacrosAuteurs") Then
       BasicLibraries.LoadLibrary("MacrosAuteurs")
    End If
    NombreCinq = NombreCinq_Implementation()
End Function
```

6) Enregistrez, fermez et ouvrez à nouveau le document Calc. Cette fois, la fonction NombreCinq() fonctionne.

Passer des arguments à une macro

Pour illustrer une fonction qui accepte des arguments, voici une macro qui calcule la somme des nombres positifs de son argument, en ignorant ceux qui sont inférieurs à zéro (Listing 5).

Listing 5. SommePositif calcule la somme des nombres positifs de son argument

```
Function SommePositif(Optional x)
  Dim LaSomme As Double
  Dim iLig as Integer
  Dim iCol as Integer

  LaSomme = 0.0
  If Not IsMissing(x) Then
    If IsArray(x) Then
       For iLig = LBound(x, 1) To UBound(x, 1)
          For iCol = LBound(x, 2) To UBound(x, 2)
             If x(iLig, iCol) > 0 Then LaSomme = LaSomme + x(iLig,
iCol)
          Next
       Next
    Else
       If x > 0 Then LaSomme = x
    End If
  End If
  SommePositif = LaSomme
End Function
```

La macro du Listing 5 permet d'illustrer des techniques importantes :

1) L'argument `x` est optionnel. Si un argument n'est pas optionnel et qu'une fonction est appelée sans lui, LibreOffice affiche un message d'avertissement à chaque fois que la macro est appelée. Si la fonction est appelée plusieurs fois, l'erreur sera alors affichée plusieurs fois.
2) `IsMissing` vérifie qu'un argument a été passé avant de l'utiliser.
3) `IsArray` vérifie si l'argument est une valeur simple ou un tableau de valeurs. Par exemple, `=SommePositif(7)` ou `=SommePositif(A4)`. Dans le premier cas, le nombre 7 est passé en tant qu'argument, et dans le second cas, la valeur de la cellule A4 est passée à la fonction.
4) Si une plage est passée à la fonction, elle est passée en tant que tableau de valeurs à deux dimensions. Par exemple, `=SommePositif(A2:B5)`. `LBound` et `UBound` servent à déterminer les limites du tableau utilisé. Bien que la limite inférieure du tableau soit 1, il est plus prudent d'utiliser `LBound` si jamais la fonction change dans le futur.
5) Le séparateur décimal est toujours le point, même si l'interface du tableur utilise la virgule.
6) Les variables `iLig` et `iCol` utilisées pour "boucler" sur la plage sont ici déclarées de type `Integer`. Ce type ne peut contenir de valeurs supérieures à 32767. Si vous devez manipuler un nombre de lignes plus grand, déclarez ces variables de type `Long`.
7) L'indentation des lignes n'est pas obligatoire, mais conseillée pour faciliter la lecture et repérer notamment les erreurs d'appariement (sub/end sub, function/end function, if/end if, etc.).

Note	La macro du Listing 5 est circonspecte et vérifie si l'argument est une valeur simple ou un tableau. Par contre, elle ne vérifie pas que chaque valeur soit numérique. Vous pouvez être aussi prudent que vous le voulez : plus vous vérifierez de choses, plus la macro sera robuste, mais plus son exécution sera lente.

Passer un ou deux arguments est tout aussi facile : ajoutez un autre argument à la définition de la fonction (Listing 6). Lorsque vous appelez dans une cellule de classeur une fonction avec deux arguments, séparez les avec un point-virgule, par exemple, `=TestMax(3;-4)`.

Listing 6. TestMax retourne le plus grand des deux arguments

```
Function TestMax(x, y)
    If x >= y Then
       TestMax = x
    Else
       TestMax = y
    End If
End Function
```

Arguments passés en tant que valeurs

Les arguments passés à une macro de Calc sont toujours des valeurs, numériques ou textuelles (jamais des objets). L'argument ne permet pas de savoir quelles cellules sont utilisées, si c'est le cas. Par exemple, `=SommePositif(A3)` passe la valeur de la cellule A3, et la macro n'a pas le moyen de savoir que la cellule A3 est utilisée. Si vous devez savoir quelles cellules sont référencées plutôt que la valeur de ces cellules, passez la plage en tant que chaîne de caractères, analysez cette chaîne et obtenez les valeurs des cellules référencées.

Astuce	Passer l'adresse de la cellule ou de la plage en tant que chaîne de caractères ("A3" par exemple) rend impossible l'adaptation relative des formules lors de la recopie dans le tableur. La solution est d'utiliser les fonctions FEUILLE et CELLULE permettant respectivement d'obtenir le numéro de feuille et l'adresse de la cellule. Exemple : =ESTRESULTPOSITIF(A1;FEUILLE();CELLULE("adresse";A1))

Accéder directement aux cellules

Vous pouvez accéder directement aux objets internes de LibreOffice pour manipuler un document Calc. Par exemple, la macro du Listing 7 additionne les valeurs de la cellule A2 de toutes les feuilles dans le document en cours. `ThisComponent` est initialisé par le langage LibreOffice Basic quand la macro démarre et référence le document en cours. Un document Calc contient des feuilles : `ThisComponent.getSheets()` retourne l'ensemble de ces feuilles. Utilisez `getCellByPosition(col, row)` pour retourner une cellule se trouvant à une ligne et une colonne déterminées.

Listing 7. Somme de la cellule A2 de toutes les feuilles

```
Function SommeFeuilles()
  Dim LaSomme As Double
  Dim i As integer
  Dim oFeuilles
  Dim oFeuille
  Dim oCellule

  oFeuilles = ThisComponent.getSheets()
  For i = 0 To oFeuilles.getCount() - 1
    oFeuille = oFeuilles.getByIndex(i)
    oCellule = oFeuille.getCellByPosition(0, 1) ' Cellule A2
    LaSomme = LaSomme + oCellule.getValue()
  Next
  SommeFeuilles = LaSomme
End Function
```

Astuce	Un objet de type cellule supporte les méthodes `getValue()`, `getString()` et `getFormula()` pour *obtenir* la valeur numérique, la valeur chaîne de caractères et la formule utilisées dans une cellule. Utilisez les méthodes "`set`" correspondantes pour *fixer* les valeurs voulues.

Utilisez `oFeuille.getCellRangeByName("A2")` pour retourner une plage de cellules à partir de son nom. Si une seule cellule est référencée, un objet cellule est alors retourné. Si une plage de cellules est fournie, alors toute la plage de cellules est retournée (voir Listing 8). Notez qu'avec une plage de cellules, la fonction retourne les données sous forme d'un tableau de tableaux, ce qui est plus lourd que de traiter un tableau à deux dimensions, comme dans le Listing 5.

Listing 8. Somme de la plage A2:C5 de toutes les feuilles

```
Function SommeFeuilles()
  Dim LaSomme As Double
  Dim iLig As Integer, iCol As Integer, i As Integer
  Dim oFeuilles, oFeuille, oCellules
  Dim oLig(), oLigs()

  oFeuilles = ThisComponent.getSheets()
  For i = 0 To oFeuilles.getCount() - 1
    oFeuille = oFeuilles.getByIndex(i)
    oCellules = oFeuille.getCellRangeByName("A2:C5")
    REM getDataArray() retourne les données comme variant,
    REM donc nombres et chaînes de caractères sont retournés.
    REM getData() retourne les données de type Double,
    REM seuls des nombres sont retournés.
    oLigs() = oCellules.getData()
    For iLig = LBound(oLigs()) To UBound(oLigs())
      oLig() = oLigs(iLig)
      For iCol = LBound(oLig()) To UBound(oLig())
        LaSomme = LaSomme + oLig(iCol)
      Next
    Next
  Next
  SommeFeuilles = LaSomme
End Function
```

Attention

Quand une macro est appelée en tant que fonction Calc, elle ne peut modifier aucune valeur dans la feuille depuis laquelle elle a été appelée.

Tri

Vous voulez trier les données de la Figure 17. Le tri se fera tout d'abord selon la colonne B en ordre décroissant, puis selon la colonne A en ordre croissant.

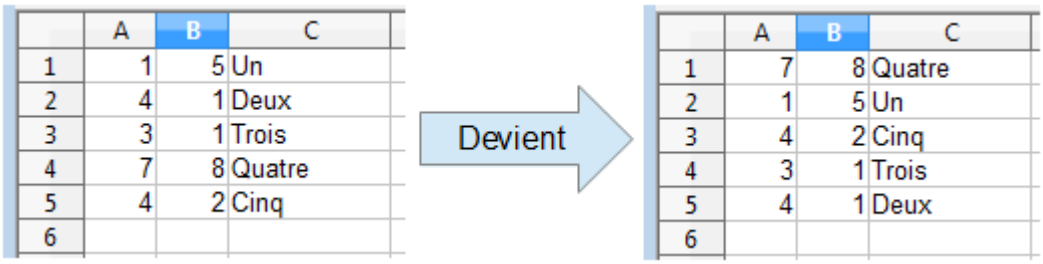


Figure 17 : Tri et son résultat

L'exemple du Listing 9 montre comment effectuer un tri sur deux colonnes.

Le tri s'applique à une plage de données. Avec un objet *Plage*, nous disposons de la méthode `Sort`. Cette méthode requiert en argument tous les paramètres de tri, sous la forme d'un « tableau de propriétés ».

L'appel à cette méthode est quant à lui très simple : `oPlage.Sort(oDescTri())`.

La difficulté est ici de créer le tableau des paramètres du tri (variable `oDescTri` dans l'exemple ci-dessus). Nous aurons des paramètres généraux (exemple : la zone de tri comprend-elle une ligne de titre), et des paramètres pour chaque clé de tri (colonne à utiliser, tri croissant ou décroissant).

Ceci correspond très exactement à ce qui est renseigné via la commande **Données > Trier** respectivement dans les onglets **Options** et **Critères de tri**.

Listing 9. Tri de la plage A1:C5 dans la Feuille1

```
Sub TriPlage
  Dim oFeuille  ' Feuille qui contient les données à trier.
  Dim oPlage    ' Plage de données à trier.

  REM Un tableau de champs de tri détermine les colonnes à
  REM trier. C'est un tableau de deux éléments, 0 et 1.
  REM Pour ne trier qu'une colonne, faites :
  REM Dim oChampsTri(0) As New com.sun.star.table.TableSortField
  Dim oChampsTri(1) As New com.sun.star.table.TableSortField

  REM Le descripteur de tri est un tableau de propriétés.
  REM La propriété primaire contient les champs de tri.
  Dim oDescTri(1) As New com.sun.star.beans.PropertyValue

  REM Retourne la feuille appelée "Feuille1"
  oFeuille = ThisComponent.Sheets.getByName("Feuille1")
  REM Retourne la plage à trier
  oPlage = oFeuille.getCellRangeByName("A1:C5")
```

```
    REM Sélectionne la plage à trier. Aurait comme
    REM effet de mettre en surbrillance les données triées.
    'ThisComponent.getCurrentController.select(oPlage)

    REM Les colonnes sont numérotées à partir de 0,
    REM la colonne A est 0, B est 1, etc.
    REM Tri de la colonne B (colonne 1) décroissant.
    oChampsTri(0).Field = 1
    oChampsTri(0).IsAscending = False

    REM Si la colonne B a deux cellules de même valeur,
    REM trier selon la colonne A croissant.
    oChampsTri(1).Field = 0
    oChampsTri(1).IsAscending = True

    REM Paramétrer le descripteur de tri.
    oDescTri(0).Name = "SortFields"
    oDescTri(0).Value = oChampsTri()
    oDescTri(1).Name = "ContainsHeader"
    oDescTri(1).Value = False

    REM Trier la plage.
    oPlage.Sort(oDescTri())
End Sub
```

Conclusion

Ce chapitre constitue un bref aperçu de la façon de créer des bibliothèques et des modules, en utilisant l'enregistreur de macro, en utilisant les macros en tant que fonctions Calc, ou en écrivant vos propres macros sans l'enregistreur. Chaque sujet mériterait au moins un chapitre, et l'écriture de vos propres macros pour Calc pourrait faire l'objet d'un ou plusieurs livres. Vous pouvez également vous reporter à la page du wiki correspondante pour des exemples supplémentaires : *http://wiki.documentfoundation.org/Macros/Calc/fr*. En d'autres termes, ceci est juste le début de ce que vous pouvez connaître !