méthode `findNext` jusqu'à échec de la recherche. Voici une recherche progressive dans toute la feuille, en explorant ligne par ligne.

```
rem CD-Rom : Code12-04.sxc   bibli : Rechercher Module3
Option Explicit

Sub TrouverToutsuccessivement()
Dim monDocument As Object, lesFeuilles As Object
Dim maFeuille As Object
Dim jeCherche As Object, posTrouve As Variant
monDocument = ThisComponent
lesFeuilles = monDocument.Sheets
maFeuille = lesFeuilles.getByName("Compositeurs")
jeCherche = maFeuille.createSearchDescriptor
with jeCherche
  .SearchString = "Otto"
  ' rechercher les cellules contenant au moins ce texte
  .SearchWords = false
  .SearchByRow = true ' rechercher par lignes
end with
posTrouve = maFeuille.findFirst(jeCherche)
Do Until isNull(posTrouve)
  posTrouve.CellBackColor = RGB(255,200,255)
  posTrouve = maFeuille.findNext(posTrouve, jeCherche)
Loop
End Sub
```

Si la recherche ne trouve rien, l'objet `posTrouve` reçoit la valeur `Null` et la boucle n'est pas exécutée. Si elle trouve, `posTrouve` est alors un objet cellule. La recherche est relancée par `findNext`, qui utilise en premier argument l'objet cellule précédemment trouvé.

Ici, nous avons forcé la recherche ligne par ligne, ce qui est visible si vous posez un point d'arrêt sur l'instruction comportant `findNext`.

Comme avec le deuxième exemple, nous pourrions limiter la recherche à une zone de cellules. D'ailleurs, elle pourrait être une zone sélectionnée par l'utilisateur, comme décrit plus haut dans la section sur les sélections visuelles.

Il est possible de sélectionner visuellement la zone trouvée, au lieu de changer la couleur de fond :

```
monDocument.CurrentController.Select(posTrouve)
```

## Rechercher pour remplacer

Lors d'une recherche progressive, la zone de texte `posTrouve` sélectionne la cellule trouvée. Nous pourrions alors utiliser cet objet cellule pour analyser le type de contenu de

la cellule, ou modifier le contenu ou le format à notre guise. Nous vous laissons faire une telle macro, à titre d'exercice.

### Tout remplacer

Dans le cas simple où toutes les occurrences doivent être systématiquement remplacées partout, les objets feuille et zone offrent la fonction `replaceAll`, qui effectue ce travail et renvoie le nombre de remplacements. Elle utilise un descripteur de remplacement, obtenu avec la méthode `createReplaceDescriptor`, qui contient tous les éléments du descripteur de recherche et ajoute `ReplaceString`, de type `String`, qui contient la chaîne de caractères à mettre à la place de celle qu'on recherche.

```
rem CD-Rom : Code12-04.sxc   bibli : Remplacer Module1
Option Explicit

Sub RemplacerPartoutdans1Feuille()
Dim monDocument As Object, lesFeuilles As Object
Dim maFeuille As Object
Dim jeCherche As Object, nbrFois As Long
monDocument = ThisComponent
lesFeuilles = monDocument.Sheets
maFeuille = lesFeuilles.getByName("Compositeurs")
jeCherche = maFeuille.createReplaceDescriptor
with jeCherche
  .SearchString = "Maurice"
  .ReplaceString = "Marcel"
  ' rechercher les cellules contenant au moins ce texte
  .SearchWords = false
end with
nbrFois = maFeuille.replaceAll(jeCherche)
print "Nombre de remplacements : " & nbrFois
End Sub
```

La macro suivante modifie toutes les cellules de style `Titre1` pour leur affecter un autre style `Résultat2`.

```
rem CD-Rom : Code12-04.sxc   bibli : Remplacer Module2
Option Explicit

Sub RemplacerStylePartoutdans1Feuille()
Dim monDocument As Object, lesFeuilles As Object
Dim maFeuille As Object
Dim jeCherche As Object
monDocument = ThisComponent
lesFeuilles = monDocument.Sheets
maFeuille = lesFeuilles.getByName("Compositeurs")
jeCherche = maFeuille.createReplaceDescriptor
```

```
with jeCherche
  .SearchString = "Titre1"
  .ReplaceString = "Résultat2"
  .SearchStyles = true
end with
maFeuille.replaceAll(jeCherche)
End Sub
```

Dans une modification de styles, le nombre de remplacements n'est pas fourni par `replaceAll`.

## Trier un tableau

**ATTENTION Compatibilité**

La version 1.1 d'OpenOffice.org a modifié le mécanisme de tri. Nous ne décrirons que ce dernier, qui ne fonctionne pas sur les versions 1.0.x ou antérieures.

Un objet zone de cellules fournit une méthode de tri (en anglais *Sort*). Nous allons effectuer un tri sur la première colonne du tableau de la feuille `Compositeurs`. Voici l'ensemble du code, que nous expliquerons progressivement.

```
rem CD-Rom : Code12-03.sxc   bibli : Calculs Module2
Option Explicit

Sub Trier1Colonne()
Dim monDocument As Object, lesFeuilles As Object
Dim maFeuille As Object, maZone As Object
Dim ConfigTri(0) As New com.sun.star.table.TableSortField
Dim DescrTri As Variant
monDocument = thisComponent
lesFeuilles = monDocument.Sheets
maFeuille = lesFeuilles.getByName("Compositeurs")
maZone = maFeuille.getCellRangeByName("A1:D118")

With ConfigTri(0)
  .Field = 0 ' colonne A = "Nom, Prénom"
  .IsAscending = true
End With

DescrTri = maZone.createSortDescriptor
setPropVal(DescrTri, "SortFields", ConfigTri())
setPropVal(DescrTri, "IsSortColumns", false)
setPropVal(DescrTri, "CopyOutputData", false)
setPropVal(DescrTri, "IsUserListEnabled", false)
setPropVal(DescrTri, "BindFormatsToContent", false)
setPropVal(DescrTri, "ContainsHeader", true)
maZone.Sort(DescrTri())
End Sub
```

```
Sub setPropVal(descr As Variant, _
               nomProp As String, valProp As Variant)
Const Titre = "Tableau de propriétés"
Dim x As Long
for x = 0 to UBound(descr)
  if descr(x).Name = nomProp then
    descr(x).Value = valProp
    Exit Sub
  end if
next
MsgBox("Propriété inconnue : " & nomProp, 16, Titre)
End Sub
```

Nous définissons la zone de tri dans le tableau, en incluant la première ligne qui contient les en-têtes de colonnes. On aurait pu l'exclure.

La méthode de tri a besoin d'un descripteur de tri (notre variable `DescrTri`), qui précise les conditions globales de tri. Parmi celles-ci, nous trouvons l'élément `SortFields`, qui contient un tableau (`Array`) de descripteurs, un par colonne à trier (notre variable `ConfigTri`). Nous allons maintenant décrire chaque descripteur.

Le descripteur de tri par colonne est une structure `com.sun.star.table.TableSortField` composée des éléments détaillés dans le tableau 12-17.

**Tableau 12–17** Descripteur de tri par colonne

| Élément | Type | Signification |
|---|---|---|
| `Field` | `Long` | Rang de la colonne dans la zone de cellules ; valeur 0 pour la première colonne. |
| `IsAscending` | `Boolean` | `True` pour un tri par ordre croissant. |
| `IsCaseSensitive` | `Boolean` | `True` pour tenir compte de la casse des caractères |
| `FieldType` | `Integer` | Type du champ. (non utilisé) |
| `CollatorLocale` | `Object` | Langue et variante nationale de langue. |
| `CollatorAlgorithm` | `String` | Méthode de tri. |

L'élément `FieldType` n'est pas utilisé car chaque cellule de tableur indique le type de donnée qu'elle contient. Le tri croissant place les cellules numériques avant les cellules de texte.

L'élément `CollatorLocale` est lui-même une structure, décrite plus haut, section « Format de nombre ». Cependant, pour les cas courants (français, anglais, espagnol, allemand...) il n'est pas nécessaire de le remplir. De même, `CollatorAlgorithm` peut être omis car il n'y a actuellement pas de choix possible (la seule valeur possible est : `alphanumeric`).