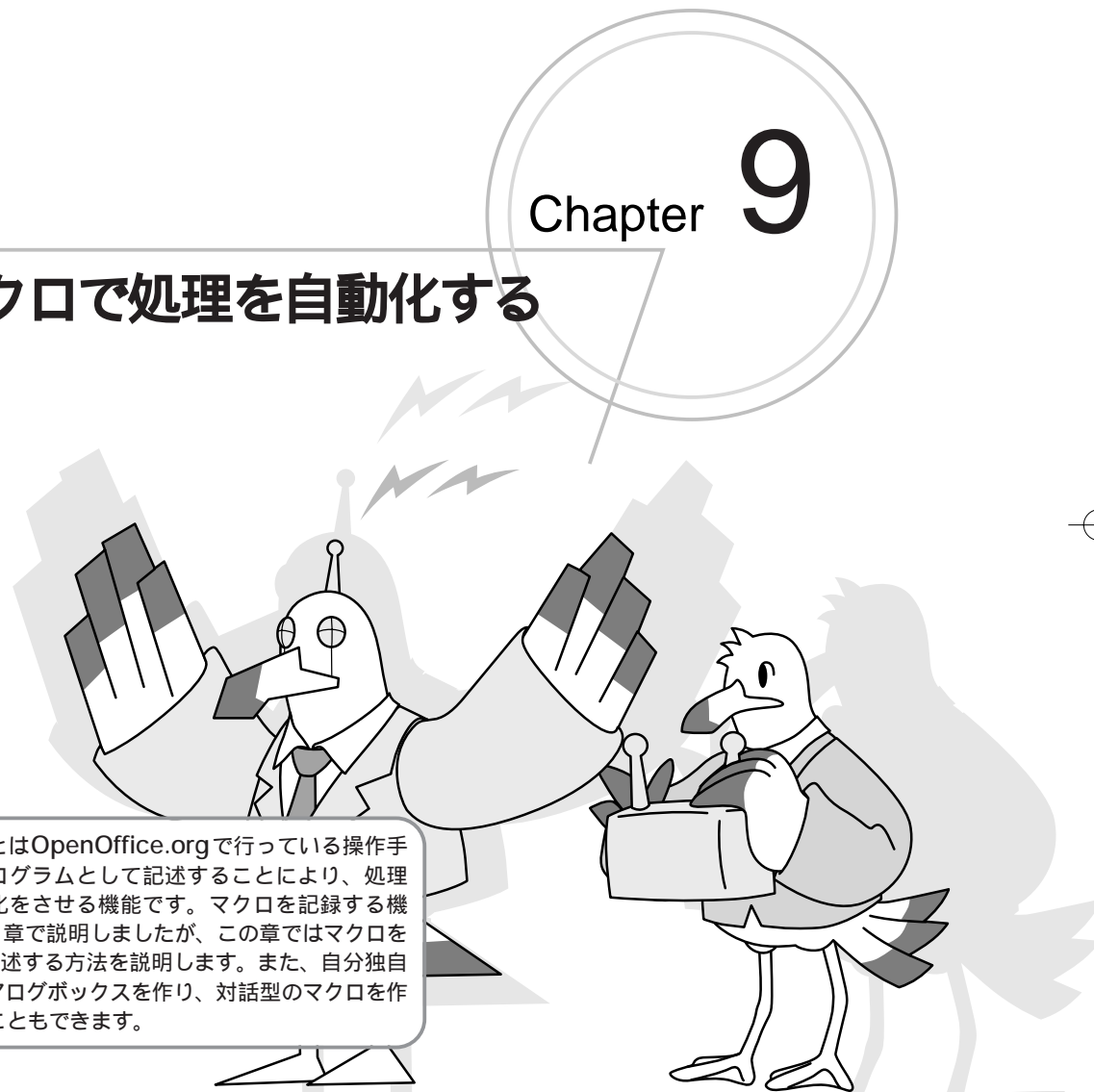


Chapter 9

マクロで処理を自動化する

マクロとはOpenOffice.orgで行っている操作手順をプログラムとして記述することにより、処理の自動化をさせる機能です。マクロを記録する機能は第3章で説明しましたが、この章ではマクロを1から記述する方法を説明します。また、自分独自のダイアログボックスを作り、対話型のマクロを作成することもできます。



マクロ入門

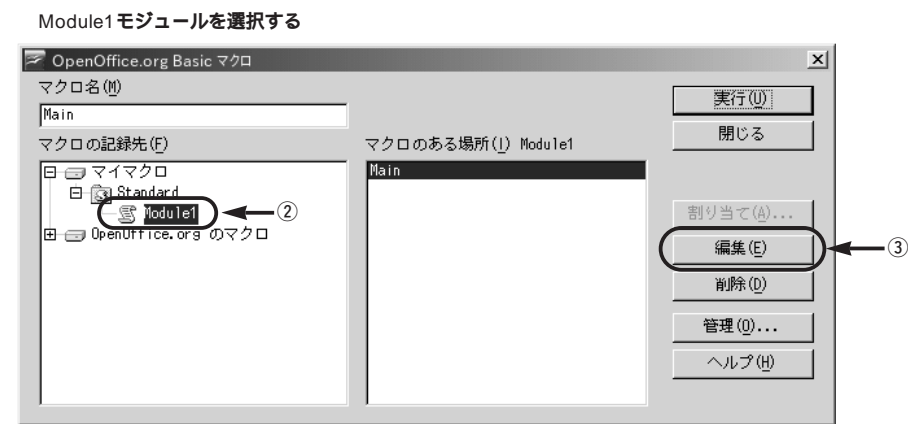
OpenOffice.org では、マクロを記述するプログラム言語として、JavaScript や BeanShell でもマクロを書くことができますが、OpenOffice.org BASIC が一番よく使われています。ここでは、OpenOffice.org BASIC の文法と OpenOffice.org BASIC により書かれたマクロやプログラムを編集したり、実行するための OpenOffice.org BASIC IDE の使い方を解説します。

マクロを作成して実行してみる

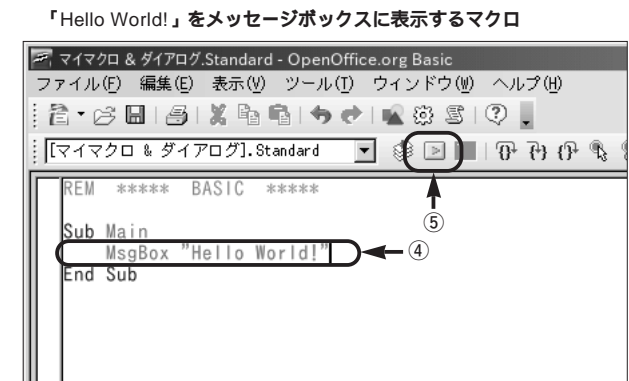
文法の説明に入る前に、マクロの実行方法を紹介しておきましょう。

では、簡単なマクロを作り、実行させてみましょう。まずは、「Hello World!」というメッセージボックスを表示するマクロを例にして、マクロの作り方の基本的な流れを学んでいきます。

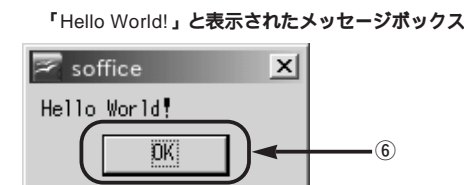
- ① メニューから [ツール (T)] [マクロ (M)] [マクロの管理 (O)] [OpenOffice.org BASIC...] を選択する
- ② 「OpenOffice.org BASIC マクロ」ダイアログボックスが表示されたら、「マクロの記録先」で「マイマクロ」「Standard」「Module1」が選択されていることを確認し、選択されていない場合には「Module1」をクリックする
- ③ **編集 (E)** をクリックする



- ④ OpenOffice.org BASIC IDE が表示されたら「Sub Main」と「End Sub」の間の空行に「MsgBox "Hello World!"」と入力する
- ⑤ **BASIC プログラムの実行** ボタンをクリックする



- ⑥ 「Hello World!」と表示されたメッセージボックスが表れ、**OK** ボタンをクリックするとマクロは終了する



このように、**BASIC プログラムの実行** ボタンをクリックすると「Sub Main」と「End Sub」の間にある命令が実行されます。

OpenOffice.org BASIC の基礎

OpenOffice.org BASIC は Visual Basic ととても似た文法を持っているため、Visual Basic または Visual Basic Application を使ったことのある人にとってはそれほど違和感なくマクロを書くことができるようになるはずです。ここでは、OpenOffice.org BASIC の数ある文法の中から、よく使う基本的なものを紹介します。

変数

変数とは、プログラム中で一時的にデータを記憶しておくための箱のようなものです。変数は宣言しなければプログラム中で使うことができません。また、変数を宣言す

る際には、型と言われる、箱の中身の種類を指定することになっています。OpenOffice.org BASICでは、変数の型には以下のようなものがあります。

データ型	扱うデータ
Integer型	- 32768 ~ 32767の整数
Long型	- 2147483648 ~ 2147483647の整数
Single型	3.402823 × 10の38乗 ~ 1.401298 × 10の - 45乗の正負の浮動小数点数
Double型	1.79769313486232 × 10の308乗 ~ 4.94065645841247 × 10の - 324乗の正負の浮動小数点数
String型	文字列
Date型	日付
Currency型	- 922337203685477.5808 ~ 922337203685477.5807の小数部4桁の数値
Boolean型	TrueまたはFalse
Object型	オブジェクト参照
Variant型	上述のすべてのデータ

これらの型の変数を宣言するには

Dim 変数名 As 型

というような文を、その変数を使用する前に記述します。たとえば、Iという名前の変数をInteger型として使用するためには、

Dim I As Integer

というようにして宣言します。

条件分岐

OpenOffice.org BASICでは条件分岐をする方法は2つあります。1つはIf...Then...Elseステートメントで、もう1つはSelect...Caseステートメントです。

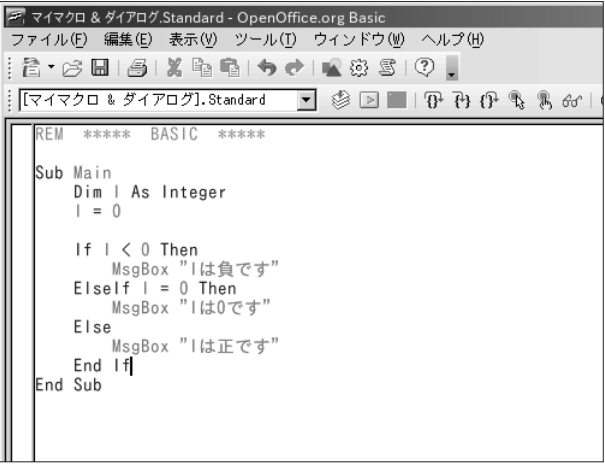
- ① 321ページで入力したソースコードのうち、「Sub Main」～「End Sub」までを次のコード1のように変更する
- ② BASIC プログラムの実行 ボタンをクリックすると「Iは0です」と書かれたメッセージボックスが表示される

Iの正負を判定する（コード1） 行頭の数字は何行目かを表すもので、入力不要

```
1: Sub Main
2:   Dim I As Integer
```

```
3:   I = 0
4:
5:   If I < 0 Then
6:     MsgBox "Iは負です"
7:   ElseIf I = 0 Then
8:     MsgBox "Iは0です"
9:   Else
10:    MsgBox "Iは正です"
11:  End If
12: End Sub
```

Iの正負を判定する



3行目の0の値を好きな整数に変えて②の操作を行ってみてください。

If...Then...Elseステートメントは、まず最初にIfとThenの間の「I < 0」を評価し、True（真）ならば次のElseIfまでの6行目が実行されます。False（偽）ならば、次にElseIfとThenの間の「I = 0」を同じように評価します。最後にどの条件に対してもFalseであるならば、ElseからEnd Ifまでの10行目が実行されます。Elseif...ThenとElseは省略することができます。

もう1つのSelect...Caseステートメントも試してみましょう。

- ① 「Sub Main」～「End Sub」までをコード2のように変更して BASIC プログラムの実行 ボタンをクリックする
- ② 3行目の「Windows」という文字列を好きな文字列に変えて BASIC プログラムの実行 ボタンをクリックして試す

Select...Caseステートメントの例（コード2）

```
1: Sub Main
2:   Dim Str As String
3:   Str = "Windows"
4:
5:   Select Case Str
6:     Case "Windows"
7:       MsgBox "Windowsです"
8:     Case "Mac"
9:       MsgBox "Macです"
10:    Case Else
11:      MsgBox "その他のOSです"
12:    End Select
13: End Sub
```

まず、Select Caseで評価の対象となる式または変数を指定します。次に、その式の結果または変数と一致するCaseから次のCaseまでが実行されます。たとえば、3行目の「Windows」を「Mac」に変えたならば、9行目が実行されます。どのCaseにも一致しない場合には、Case ElseからEnd Selectの間の11行目が実行されます。

繰り返し処理

繰り返し処理を行う方法にも2つあります。1つはFor...Nextステートメントです。これは、ある回数だけ繰り返し実行を行いたい場合によく使われます。

- ① 「Sub Main」～「End Sub」までをコード3のように変更して、**BASIC プログラムの実行** ボタンをクリックする
- ② 4行目の「5」という数値を好きな整数に変えてプログラムを実行して変化を確認する

For...Nextステートメントの例（コード3）

```
1: Sub Main
2:   Dim I As Integer
3:
4:   For I = 1 To 5
5:     MsgBox I
6:   Next I
7: End Sub
```

まずループカウンタを指定する必要があります。コード3ではIです。次に、このループカウンタの取りうる範囲を指定します。コード3ではIが1から5までの値を取ります。ForからNextまでの間が指定回数分だけ実行されるので、コード3では5行目が5回繰り返

返し実行されます。

もう1つの方法は、Do...Loopステートメントです。While...を評価し、Trueである間はLoopまでを繰り返し実行し続けます。

- ① 「Sub Main」～「End Sub」までをコード4のように変更して **BASIC プログラムの実行** ボタンをクリックする
- ② 3行目の0という数値を好きな整数に変えてプログラムを実行し、変化を確認する

Do While...Loopステートメントの例（コード4）

```
1: Sub Main
2:   Dim I As Integer
3:   I = 0
4:
5:   Do While I < 5
6:     MsgBox I
7:     I = I + 1
8:   Loop
9: End Sub
```

また、コード5もコード4と同様ですが、最低でも1回はDoとLoopの間が実行されます。つまり、3行目の0を5に変えた場合には、コード4の繰り返し処理は一度も実行されませんが、コード5では1回だけ実行されます。

Do...Loop Whileステートメントの例（コード5）

```
1: Sub Main
2:   Dim I As Integer
3:   I = 0
4:
5:   Do
6:     MsgBox I
7:     I = I + 1
8:   Loop While I < 5
9: End Sub
```

プロシージャと関数

プログラムの一部をプロシージャや関数としてひとまとまりにしておくと、プログラムを機能別に整理することができます。また、それらを別のプログラムで再利用することもできるようになります。

- ① 現在表示されているソースコードをコード6のように書き換える
- ② BASIC プログラムの実行 ボタンをクリックすると「100」と書かれたメッセージボックスが表示される

3つの整数のうち最大値をメッセージボックスに表示するプロシージャ（コード6）

```
1: Sub Main
2:   ShowMaximalValue(1,10,100)
3: End Sub
4:
5: Sub ShowMaximalValue(A As Integer, B As Integer, C As Integer)
6:   If A > B Then
7:     If A > C Then
8:       MsgBox A
9:     Else
10:      MsgBox C
11:    EndIf
12:   Else
13:     If B > C Then
14:       MsgBox B
15:     Else
16:       MsgBox C
17:     EndIf
18:   EndIf
19: End Sub
```

2行目の1、10、100といった数値を適当な整数に変えて同じように実行してみてください。

コード6はプロシージャの書き方とその使用例です。ShowMaximalValueプロシージャは3つの整数を引数として受け取り、そのうちの最大値をメッセージボックスに表示します。Subのあとに続くShowMaximalValueはプロシージャ名といわれプログラム中でこのプロシージャを呼び出すのに使用します。括弧内は引数の指定です。引数にも引数名と型があり、変数と同じようにして宣言します。プロシージャはSubからEnd Subまでが実際の処理コードになります。

関数もプロシージャと同じようなものですが、戻り値を返すことができるのがプロシージャと異なります。

3つの整数のうち最大値を返す関数（コード7）

```
1: Sub Main
2:   Dim I As Integer
3:   I = MaximalValue(1,10,100)
```

```
4:   MsgBox I
5: End Sub
6:
7: Function MaximalValue(A As Integer, B As Integer, C As Integer)
8:   As Integer
9:   If A > B Then
10:    If A > C Then
11:      MaximalValue = A
12:    Else
13:      MaximalValue = C
14:    EndIf
15:  Else
16:    If B > C Then
17:      MaximalValue = B
18:    Else
19:      MaximalValue = C
20:    EndIf
21: End Function
```

関数を定義するには、Subの代わりにFunctionを使用します。Functionに続くMaximalValueが関数名になります。戻り値を返すにはこの関数名に返したい値を代入するようにします。また、戻り値にも型があり、引数の指定したあとに「As Integer」のようにして戻り値型を指定します。

マクロの実行とデバッグ

ここでは、マクロの実行とデバッグの方法について学びましょう。OpenOffice.org BASIC IDEにはマクロを簡単にデバッグする一通りの機能が揃っています。これを利用してみましょう。

デバッグするマクロの準備

まず、表示されているすべてのコードを消し、コード8のように記述します。


デバッグを行うためのマクロ（コード8）


```
1: Sub Main
2:   Dim I As Integer
3:   Dim J As Integer
4:   Dim K As Integer
5:   Dim Ret As Integer
6:
```




```
7: I = 10
8: J = 20
9: K = 30
10: MsgBox "最大値は"
11: Ret = MaximalValue(I,J,K)
12: MsgBox Ret
13: End Sub
14:
15: Function MaximalValue(A As Integer, B As Integer, C As Integer)
    As Integer
16:   If A > B Then
17:     If A > C Then
18:       MaximalValue = A
19:     Else
20:       MaximalValue = C
21:     EndIf
22:   Else
23:     If B > C Then
24:       MaximalValue = B
25:     Else
26:       MaximalValue = C
27:     EndIf
28:   EndIf
29: End Function
```


実行と中断

マクロを実行するには、「マクロ」ツールバーの「BASIC プログラムの実行」ボタン () をクリックするか、**F5** キーを押します。これにより、現在表示しているコードの一番最初のプロシージャまたは関数が実行されます。コード8ではMaximalValue関数よりもMainプロシージャが先に記述されているので、Mainプロシージャが最初の実行され、30と表示されたメッセージボックスが現われます。

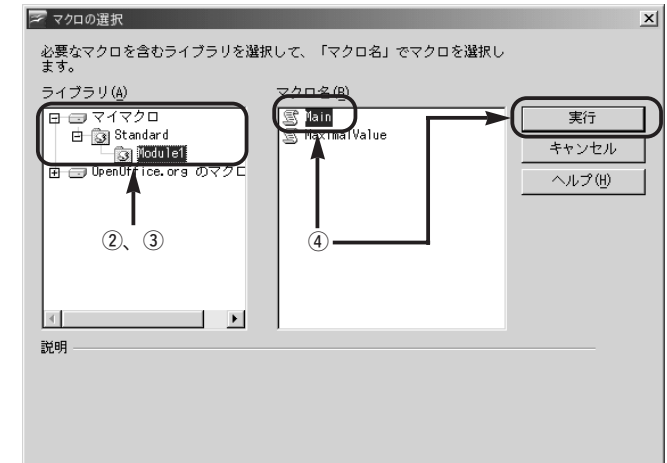
繰り返し処理が無限ループにおちいつてしまったなどの場合には、マクロを途中で中断させる必要があります。「マクロの実行をストップ」ボタン () をクリックするか、**Shift** + **F5** キーを押します。これ以降、マクロの実行には**F5** キーを、マクロの中断には**Shift** + **F5** キーを使用しましょう。

OpenOffice.org BASIC IDEを起動せずに、また、二番目以降に記述されているプロシージャや関数を実行させるには、次のように操作します。

- ① メニューから [ツール (T)] [マクロ (M)] [マクロを実行 (U)] を選択する
- ② 「マクロの選択」ダイアログボックスが表示されたら、「ライブラリ」リストから「マイマクロ」の左側の  ボタンをクリックする

- ③ さらに「Standard」の左側にある  ボタンをクリックし、「Module1」をクリックする
- ④ [マクロ名 (B)] リストの中からMainをクリックし、**実行** ボタンをクリックする
- ⑤ Mainプロシージャが実行され、30と表示されたメッセージボックスが現われる

「マクロの選択」ダイアログボックスからマクロを実行する




ただし、ここで起動できるプロシージャや関数は引数をとることはできません。MaximalValue関数などの引数をとるプロシージャや関数を実行しようとするとエラーが出ます。

引数をとる関数を実行しようとした時のエラー



ブレークポイントを設定する

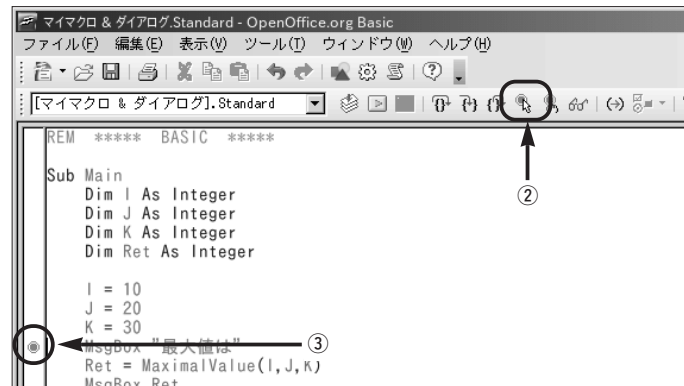
マクロの実行をあるところで一時的に中断させたいときには、ブレークポイントを使用します。

- ① 10行目をクリックし、カーソルを10行目に置く
- ② 「マクロ」ツールバーの「ブレークポイント オン/オフ」ボタン () をクリックするか、**F9** キーを押す
- ③ 10行目の左側にブレークポイントを設定したことを示した、赤丸のマークが表示される

9

マクロで処理を自動化する

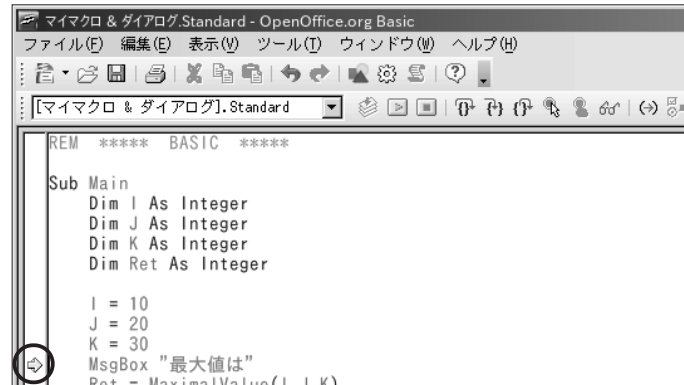
ブレークポイントを設定した



④ **F5** キーを押してマクロを実行する


⑤ 10行目を実行する直前で一時的に処理が中断し、処理が中断している行の左側に黄色い矢印が表示される

ブレークポイントによる処理の一時中断



⑥ **F5** キー押すと処理が再開し、完全に処理を中断させたい場合には **Shift** + **F5** キーを押す

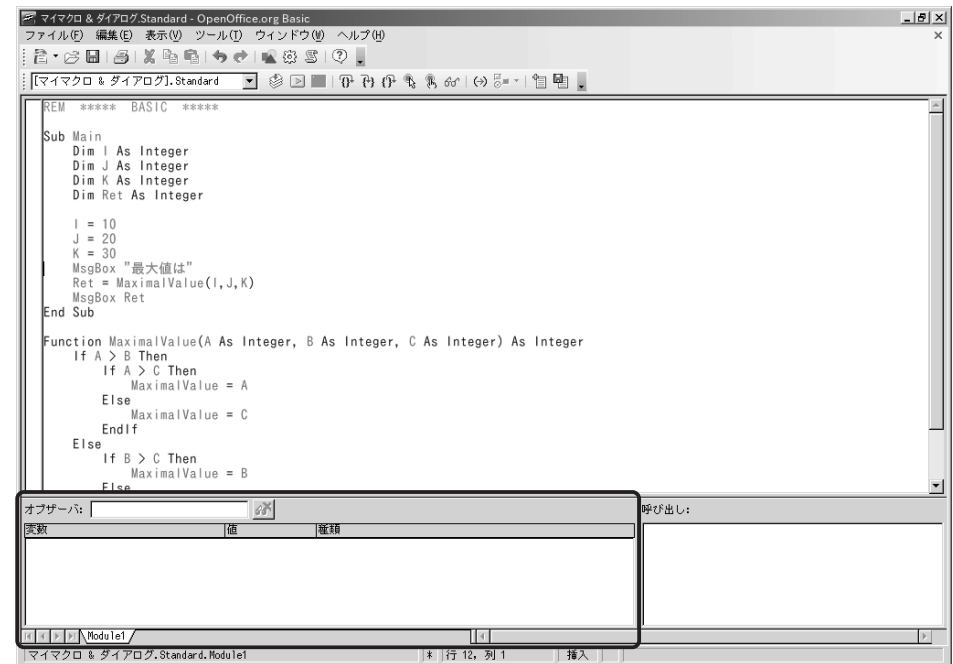
ブレークポイントを解除するには、以下のように操作します。

- ① 10行目をクリックし、カーソルを10行目に置く
- ② **ブレークポイント オン/オフ** ボタン () をクリックするか **F9** キーを押すと10行目の左側の赤丸マークが消える


○ オブザーバで変数を監視する

オブザーバを使用すると、実行中のマクロで使われている変数やオブジェクトの監視と監視中の変数の値の変更ができます。「オブザーバ」ウィンドウはOpenOffice.org BASIC IDEのソースコードの編集ウィンドウの左下側にあります。

「オブザーバ」ウィンドウ



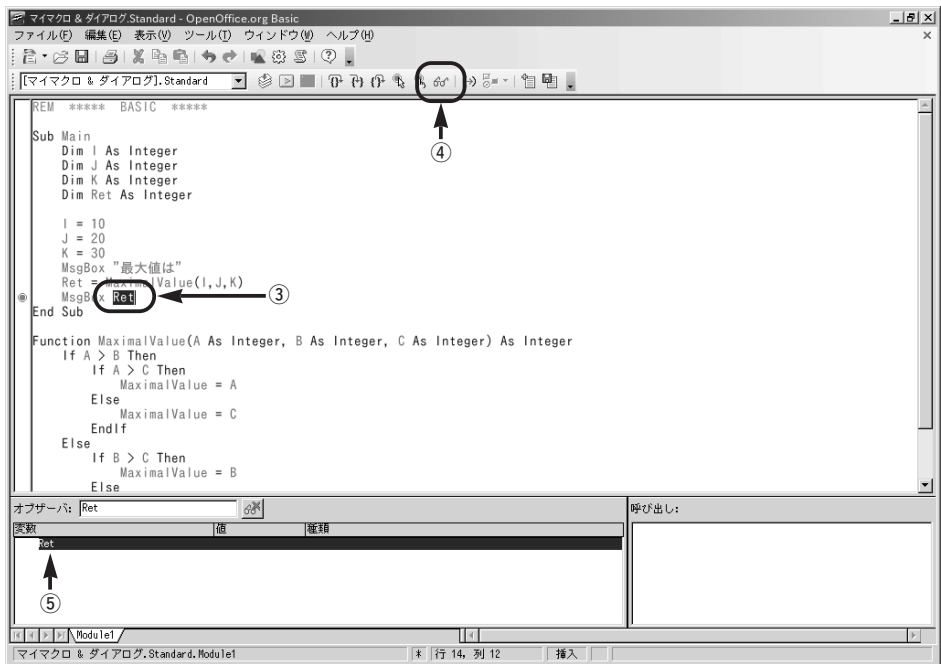
実際に変数の変化を監視してみましょう。

- ① 12行目をクリックし、カーソルを12行目に置く
- ② **F9** キーを押して12行目にブレークポイントを設定する
- ③ 12行目のRetをドラッグして選択する
- ④ 「マクロ」ツールバーの **オブザーバ オン/オフ** ボタン () をクリックするか、 **F7** キーを押す
- ⑤ 「オブザーバ」ウィンドウの「変数」列にRetと表示された行が追加される

9

マクロで処理を自動化する

変数をオブザーバへ追加する



- ⑥ **F5** キーを押してマクロを実行する
- ⑦ 12行目を実行する直前で処理が一時中断する
- ⑧ 「オブザーバ」ウィンドウで変数の状態が確認できる

「オブザーバ」ウィンドウのRet行で、変数の中身が表示される「値」列は「30」、変数の型が表示される「種類」列は「Integer」となっています。

オブザーバによる変数の監視

オブザーバ: Ret		
変数	値	種類
Ret	30	Integer

- ⑨ **F5** キーを押して12行目以降のマクロを実行すると、メッセージボックスには30が表示される

「オブザーバ」ウィンドウを使えば、一時中断したマクロに使われている変数の中身を変更することができます。


- ① **F5** キーを押してマクロを実行する
- ② 12行目を実行する直前で処理が一時中断する
- ③ 「オブザーバ」ウィンドウのRet行の「値」列（30と表示されているところ）をクリックすると、「30」という値が編集できるようになる

オブザーバによる変数の中身の変更

オブザーバ: Ret		
変数	値	種類
Ret	30	Integer

- ④ この値を「40」に変更して **Enter** キーを押す
- ⑤ **F5** キーを押してマクロの実行を続けると、メッセージボックスには「40」が表示される

なお、選択している変数の監視を止めるには、以下のように操作します。

- ① 「オブザーバ」ウィンドウからRet行をクリックする
- ② オブザーバの削除 ボタン () をクリックする

モジュールとライブラリ

いくつかのプロシージャや関数をまとめて管理することができるととても便利です。OpenOffice.orgにはプロシージャや関数をまとめて管理するための機能が備わっています。いくつかのプロシージャや関数をまとめたものをモジュールと呼び、さらにいくつかのモジュールをまとめたものをライブラリと呼びます。OpenOffice.orgでは、このモジュールとライブラリにマクロを分類、整理、管理することができます。

ライブラリの管理

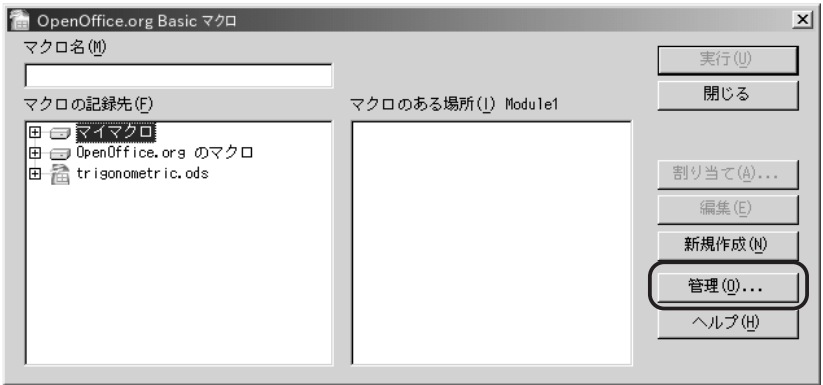
まず、モジュールとライブラリの管理を行う「マクロの管理」ダイアログボックスの使い方を説明します。

- ① Calcを起動する
- ② メニューから [ファイル(F)] [名前をつけて保存(A)] を選択し、「trigonometric.ods」

として保存する

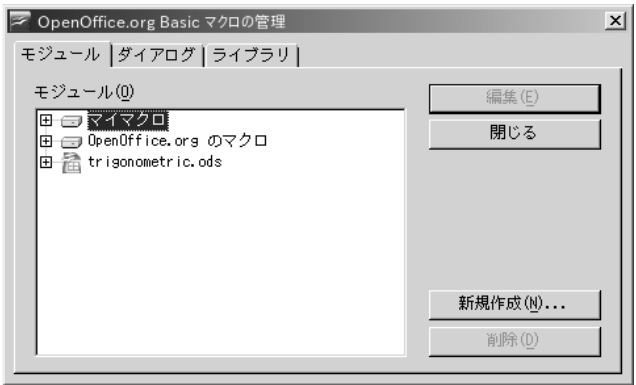
- ③ メニューから [ツール (T)] [マクロ (M)] [マクロの管理 (O)] [OpenOffice.org BASIC] を選択する
- ④ 「OpenOffice.org BASIC マクロ」ダイアログボックスが表示されたら、**管理 (O)** ボタンをクリックする

「マクロ」ダイアログボックス



- ⑤ 「マクロの管理」ダイアログボックスが表示される

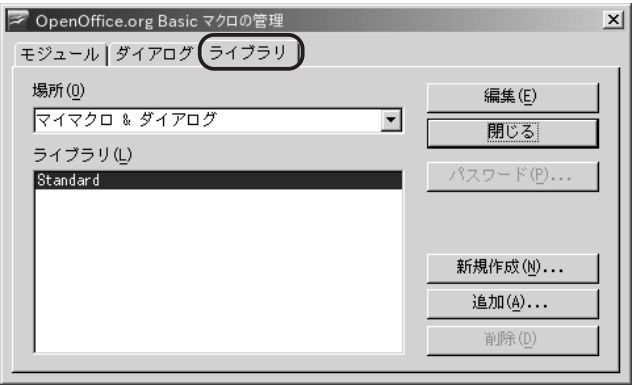
「マクロの管理」ダイアログボックス



このダイアログボックスでは、モジュールとライブラリの管理ができます。

- ① 「ライブラリ」タブをクリックする

ライブラリの管理

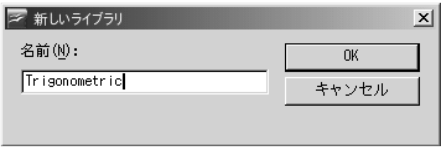


「場所」には現在どのファイルに添付されているライブラリを管理しているのかが表示されます。「マイマクロ & ダイアログ」は特別な場所で、使用しているOSにインストールされているOpenOffice.orgからいつでも利用できるライブラリを保存しておく場所です。また、「OpenOffice.orgのマクロ & ダイアログ」も特別な場所で、OpenOffice.orgをインストールする際にあらかじめ付いてくるライブラリです。この場所に保存されているライブラリは同じバージョンのOpenOffice.orgならば別のコンピュータにインストールされたOpenOffice.orgからでも共通して利用することができます。

さて、ここでは先ほど保存したtrigonometric.odsに「Trigonometric」というライブラリを作成することにしましょう。ファイルにライブラリを添付することにより、作成したマクロをドキュメントと一緒に他人に送ったりすることができるようになります。

- ① 「場所」で「trigonometric.ods」を選択する
- ② **新規作成 (N)** ボタンをクリックする
- ③ 「新しいライブラリ」ダイアログボックスの「名前」の欄に「Trigonometric」と入力して **OK** ボタンをクリックする

「新しいライブラリ」ダイアログボックス

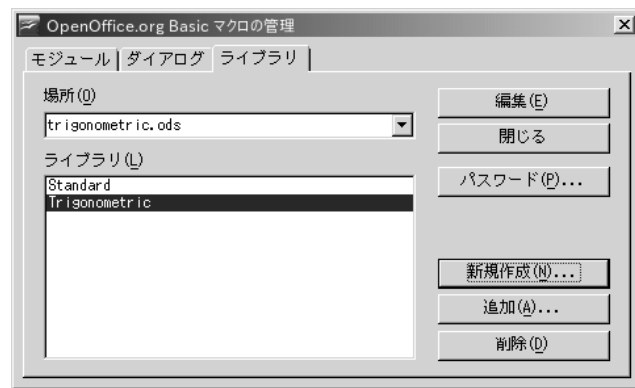


- ④ 「Trigonometric」というライブラリが、trigonometric.odsに追加される

9

マクロで処理を自動化する

Trigonometricライブラリを追加した結果

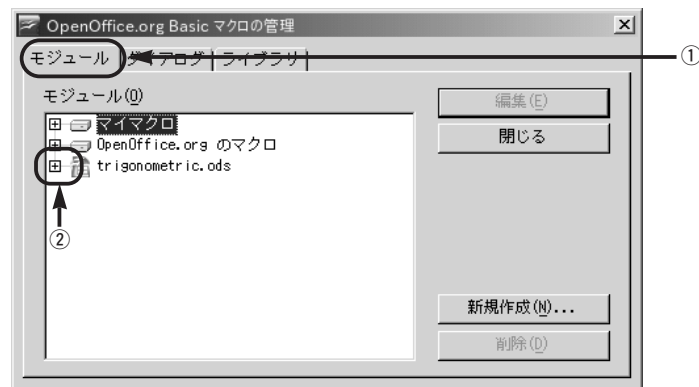


モジュールの管理

次に「Trigonometric」ライブラリに「TriFunc」というモジュールを作成してみましょう。

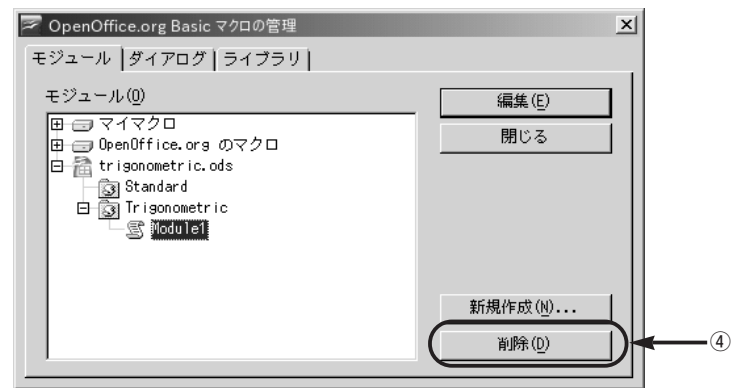
- ① 「モジュール」タブをクリックして表示する
- ② 「モジュール」のリスト中のtrigonometric.odsの左側にある **+** をクリックしてtrigonometric.odsに添付されているライブラリを表示する

モジュールの管理



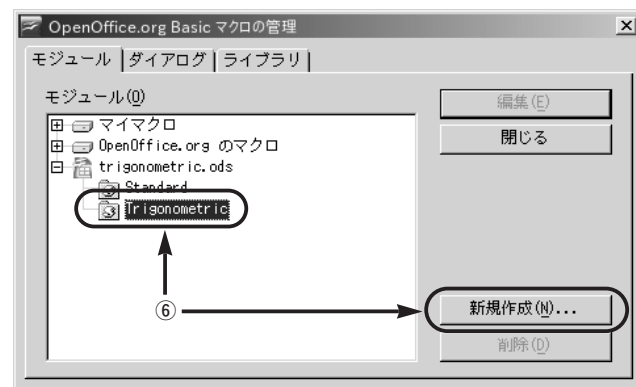
- ③ 「Trigonometric」の左側にある **+** をクリックすると、すでに「Module1」というモジュールが「Trigonometric」ライブラリにあることが分かる
- ④ 「Module1」をクリックしてから **削除 (D)** ボタンをクリックする

Trigonometricに含まれるモジュール



- ⑤ 確認のダイアログボックスが表示されたら、 **はい (Y)** ボタンをクリックして「Module1」モジュールを削除する
- ⑥ 「Trigonometric」を選択し、 **新規作成 (N)** ボタンをクリックする

新しいモジュールの追加

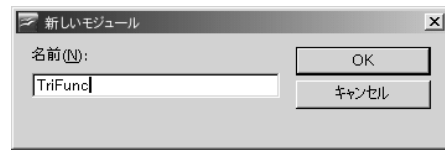


- ⑦ 「新しいモジュール」ダイアログボックスの「名前」欄に「TriFunc」と入力して **OK** ボタンをクリックする

9

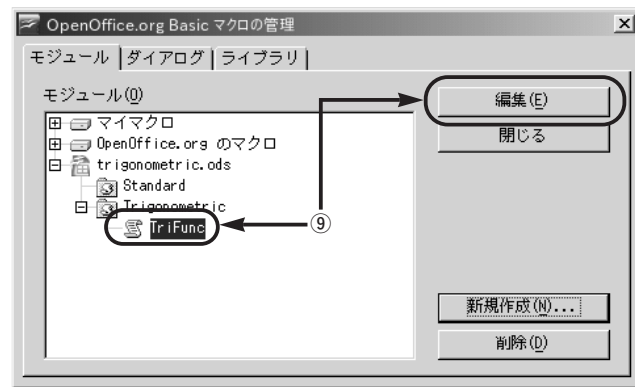
マクロで処理を自動化する

「新しいモジュール」ダイアログボックス



- ⑧ 「Trigonometric」ライブラリに「TriFunc」モジュールが追加された
- ⑨ 「TriFunc」を選択し **編集(E)** ボタンをクリックするとOpenOffice.org BASIC IDEが起動し、「TriFunc」モジュールを編集できる

「TriFunc」モジュールが追加された



また、すでに存在するモジュールについては、OpenOffice.org BASIC IDEで編集するモジュールを切替えることができます。「マクロ」ツールバーの一番左側にあるライブラリリストボックスの中からライブラリを選択すると、そのライブラリに所属するモジュールがすべて開かれます。そのライブラリのあるモジュールを編集したいのであれば、OpenOffice.org BASIC IDEの一番下にあるタブリストの中から編集したいモジュール名をクリックすると、そのモジュールを編集できるようになります。

OpenOffice.org BASIC IDEでのライブラリとモジュールの切り替え



ダイアログを使ったマクロ

自分でダイアログを作成し、それをマクロで使うことができます。ダイアログを使えば、対話式的マクロが作成できるようになり、マクロの利用の幅が大きく広がります。ぜひ、このダイアログを使ったマクロを作成する方法をマスターし、OpenOffice.orgの操作を楽にしましょう。

ダイアログエディタの使用方法

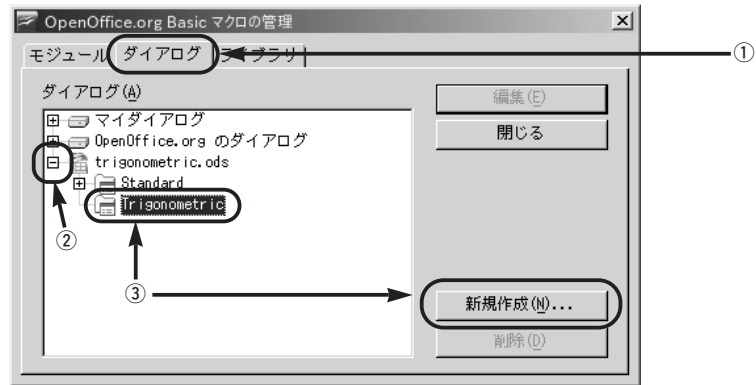
OpenOffice.org BASIC IDEにはダイアログを視覚的かつ簡単に作成するためのダイアログエディタが備わっています。これを使用すれば、Drawで図を書くような感じで誰にでもダイアログを作成することができます。

新しいダイアログボックスを作成する

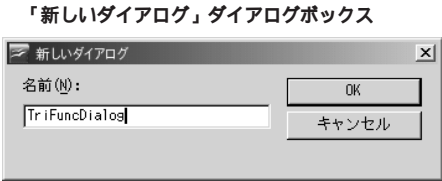
まずは、ダイアログそのものを作成します。

- ① メニューから **【ツール(T)】** **【マクロ(M)】** **【ダイアログの管理(D)】** をクリックすると「OpenOffice.org マクロの管理」ダイアログボックスが「ダイアログ」タブを開いた状態で表示される
- ② 「trigonometric.ods」の左側にある **+** をクリックする
- ③ 「Trigonometric」を選択し、 **新規作成(N)...** ボタンをクリックする

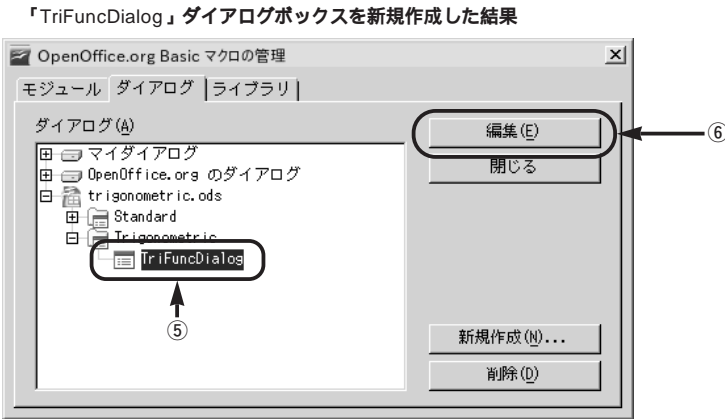
ダイアログボックスの新規作成



- ④ 【新しいダイアログ】ダイアログボックスの「名前」欄に「TriFuncDialog」と入力して **OK** ボタンをクリックする



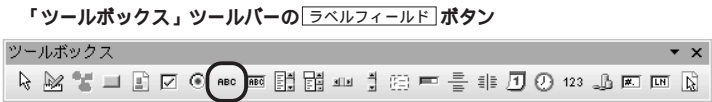
- ⑤ 「TriFuncDialog」という項目が追加された
- ⑥ 「TriFuncDialog」をクリックし、**編集 (E)** ボタンをクリックすると、ダイアログエディタが起動する



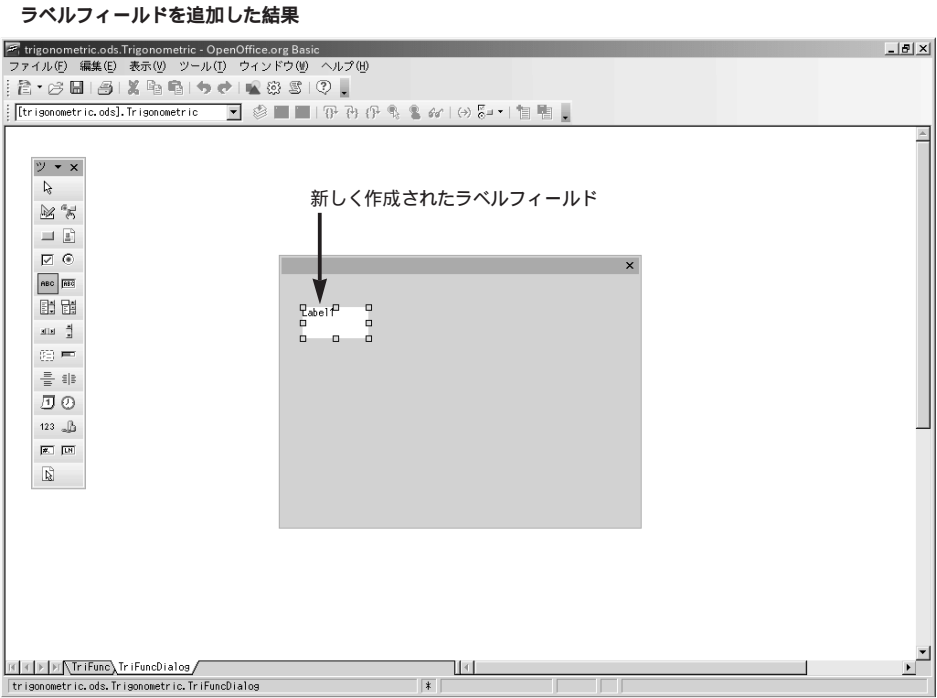
ダイアログへコントロールを配置する

ダイアログにコントロールを配置してみましょう。コントロールとはダイアログボックス上のユーザーインターフェースの構成要素であり、キーボードからの入力を受け付けるテキストフィールドや、マウスでつまみを移動させることにより値を変更できるスクロールバーなどがあります。これらのコントロールは好き勝手にダイアログ上に配置することができます。

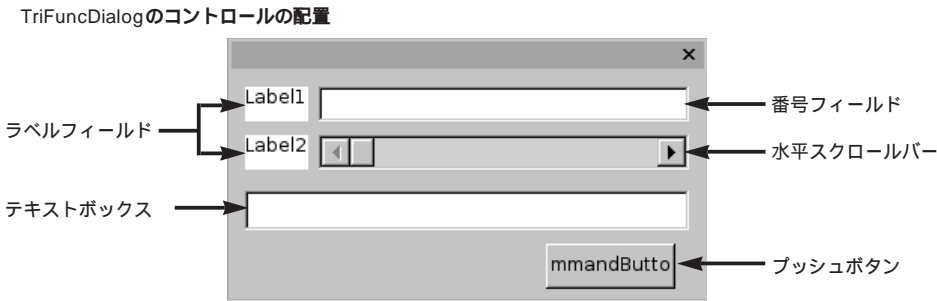
- ① 「ツールボックス」ツールバーから **ラベルフィールド** ボタンをクリックする



- ② ダイアログ上のラベルフィールドを配置したい場所の左上から右下までをドラッグする



- ③ 同じようにして、「ツールボックス」ツールバーで配置したいコントロールのボタンをクリックし、図のようにダイアログ上に配置する



一度配置してしまったコントロールの大きさや位置は、あとから変更することができます。位置を変更するには、コントロールをクリックして移動したい場所までドラッグし、そのコントロール以外のところをクリックして選択を解除します。

また、コントロールの大きさの変更は、コントロールをクリックして選択すると、コントロールの周りに緑四角印が表示されるので、この緑四角印を上下左右にドラッグすることにより大きさを変更します。大きさを変更したら、そのコントロール以外のところをクリックします。

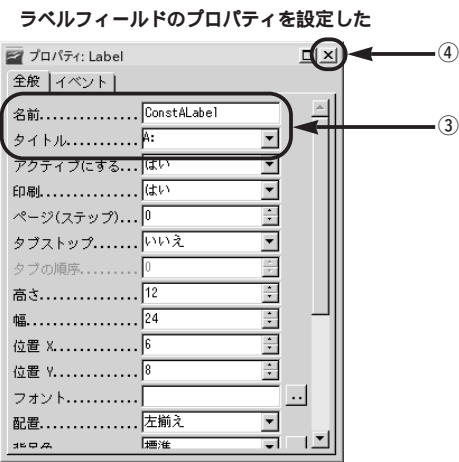
ダイアログの縁をクリックすると、ダイアログ自体の大きさを変更できます。ダイアログの周りに同様に緑四角印が表示されるので、これを上下左右にドラッグすることにより大きさを変更します。

コントロールのプロパティを変更する

ダイアログ上に配置したコントロールはそれぞれプロパティを持っています。プロパティとは、そのコントロールの属性を決めるものです。このプロパティを変更することで、コントロールに表示される文字列や背景色を設定することができます。

一番左上に配置したラベルの「名前」と「タイトル」を変更してみましょう。

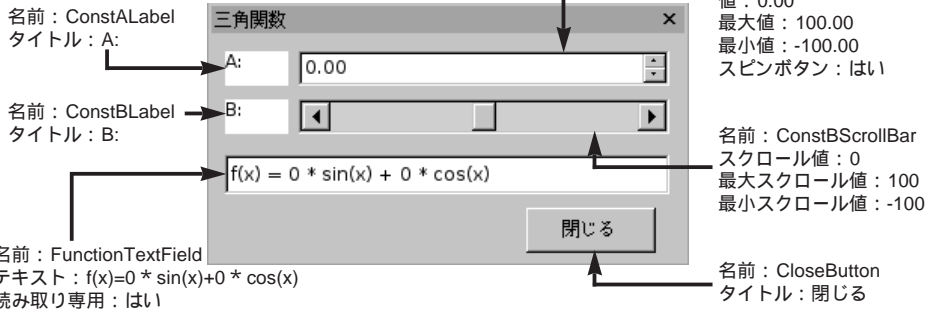
- ① 一番左上に配置したラベルフィールドをダブルクリックする
- ② 「プロパティ : Label」ダイアログボックスが表示される
- ③ 「名前」を「LabelConstA」, 「タイトル」を「A:」に変更する
- ④ ダイアログボックスの右上の「x」ボタンをクリックして閉じる



コントロールに表示されている文字列が「A:」に変わりました。ちなみに、名前プロパティの値（文字列）はマクロで使うときに必要になります。

同じようにして、他のコントロールのプロパティも図のように設定してください。また、ダイアログのプロパティを変更するには、ダイアログボックスの縁をダブルクリックします。

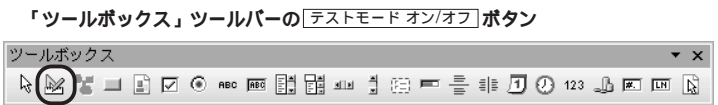
各コントロールのプロパティ



テストモード

作成したダイアログボックスは、テストモードにより実際にどのようにダイアログが表示されるかを確認することができます。

- ① 「ツールボックス」ツールバーの「テストモード オン/オフ」ボタンをクリックする



- ② ダイアログ上のコントロールを動作させて、実際の動作を確認する
- ③ ダイアログの右上の「x」ボタンをクリックするとテストモードが終了する

ダイアログを使用したマクロの実行

ダイアログを作成しただけでは何も機能しません。マクロから作成したダイアログを表示するように命令し、さらにダイアログに配置されたコントロールの動作に応じてマクロが実行されるようにしなければいけません。

ダイアログを表示するためのプロシージャを用意する

まずはダイアログを表示させるためのプロシージャが必要です。

- ① OpenOffice.org BASIC IDE の下のタブの中から「TriFunc」を選択する

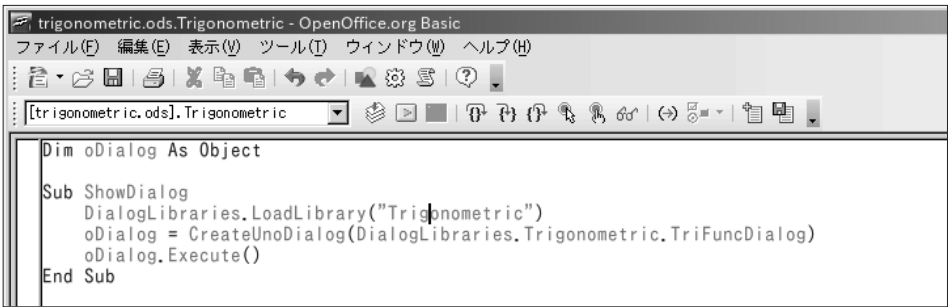


- ② 「TriFunc」モジュールの編集ウィンドウに切り替わる
- ③ 「TriFunc」モジュール全体を以下のコード9に書き換える
- ④ **[F5]** キーを押してマクロを実行すると作成したダイアログが表示される
- ⑤ このマクロを終了させるには、ダイアログの**[x]** ボタンをクリックする

ダイアログを表示させるプロシージャ（コード9）

```
1: Dim oDialog As Object
2:
3: Sub ShowDialog
4:   DialogLibraries.LoadLibrary("Trigonometric")
5:   oDialog = CreateUnoDialog(DialogLibraries.Trigonometric.
        TriFuncDialog)
6:   oDialog.Execute()
7: End Sub
```

編集後のTriFuncモジュール



コード9はダイアログを表示させるためのプロシージャと変数宣言です。1行目でSub ~ End Subの外で変数を宣言しているのは、パブリックドメイン変数と呼ばれるものです。パブリックドメイン変数は同じライブラリ内にあるプロシージャまたは関数で1つの変数を共有することができます。

4行目のLoadLibraryメソッドの引数には、表示させたいダイアログが所属するライブラリ名を指定します。CreateUnoDialog関数にはDialogLibraries.ライブラリ名.ダイアログボックス名という引数を与えることにより、そのダイアログを新しく作成しそのダイアログのオブジェクト参照を返します。最後に6行目にあるExecuteメソッドを実行するとダイアログが表示されます。

コントロールへのイベントの割り当て

たとえば、ボタンを押したときにダイアログを閉じるようにしたいならば、まず、ダイアログを閉じるためのプロシージャを書き、さらにボタンを押したときにそのプロシ

ージャを実行するようにイベントを割り当てなければいけません。

まず、コード10をTriFuncモジュールに追加します。CloseDialogプロシージャがダイアログを閉じるためのプロシージャになります。

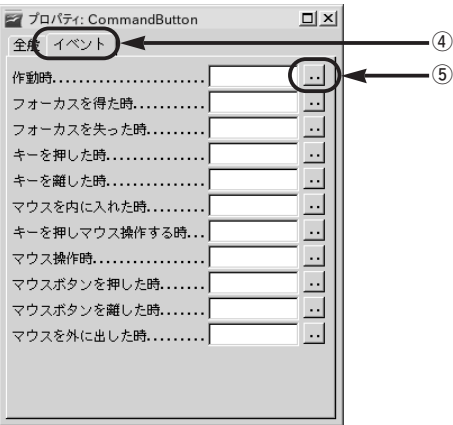
- ① コード10の3行をTriFuncモジュールに追記する

ダイアログを閉じるプロシージャ（コード10）

```
1: Sub CloseDialog
2:   oDialog.EndExecute()
3: End Sub
```

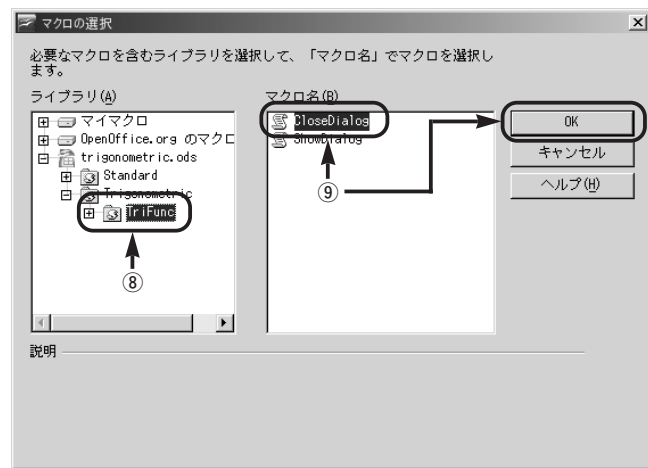
- ② OpenOffice.org BASIC IDE の下の「TriFuncDialog」タブをクリックする
- ③ **ダイアログ編集ウィンドウに切り替わるので** **[閉じる]** ボタンコントロールをダブルクリックして「プロパティ: CommandButton」ダイアログボックスを表示する
- ④ 「イベント」タブをクリックして表示する
- ⑤ 「作動時」の右側にある **[...]** ボタンをクリックする

「イベント」タブのプロパティ



- ⑥ 「マクロの割り当て」ダイアログボックスが表示されるので、**[割り当て (A)]** ボタンをクリックする
- ⑦ 「マクロの選択」ダイアログボックスが表示されたら「ライブラリ」リストから「trigonometric.ods」の左側にある**[+]** ボタンをクリックする
- ⑧ さらに「Trigometric」の左側にある**[+]** ボタンをクリックし、「TriFunc」をクリックする
- ⑨ 「マクロ名」リストの中から「CloseDialog」をクリックし、**[OK]** をクリックする

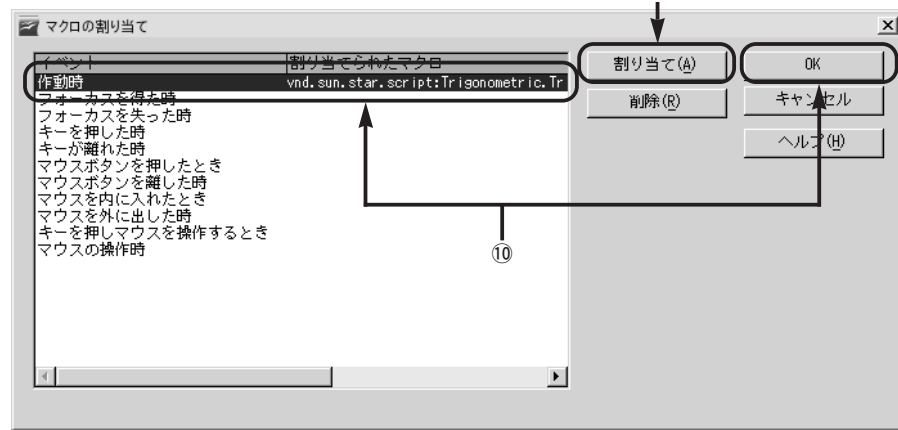
「マクロの選択」ダイアログボックスで「CloseDialog」を選択する



これで、このボタンがクリックされたときにCloseDialogプロシージャが実行されるようになりました。

- ⑩ ボタンをクリックして「マクロの割り当て」ダイアログボックスを閉じる

マクロの割り当てダイアログボックス



- ⑪ 「プロパティ : CommandButton」ダイアログボックスも右上の をクリックして閉じる
- ⑫ OpenOffice.org BASIC IDE の下の「TriFunc」タブをクリックし、TriFuncモジュールのコードの編集ウィンドウに戻る
- ⑬ キーでShowDialog プロシージャを実行する
- ⑭ ボタンでダイアログが閉じ、マクロが終了することを確認する

ダイアログを閉じるには、ダイアログオブジェクトのEndExecuteメソッドを呼び出すだけです。



マクロによるコントロールの操作とプロパティの取得

コントロールに入力された値やコントロールの状態（プロパティ）をマクロで使用したり、あるいはマクロからコントロールに値を入力してみたりコントロールのプロパティを変更することができます。その方法を学びましょう。

- ① コード11をTriFuncモジュールに追加する
- ② ボタンの「開始時」イベントにCloseDialog プロシージャを割り当てたのと同じようにして、「番号」フィールドの「テキストを変更した時」イベントと「ConstBScrollBar」の「調整する時」イベントにそれぞれ「ReCalculate」プロシージャを割り当てる
- ③ キーを押してShowDialog プロシージャを実行する
- ④ 「番号」フィールドの値を変更してみたり、スクロールバーを動かしてみると、それに応じてテキストボックスに関数が表示される
- ⑤ ボタンのクリックでダイアログボックスが閉じ、マクロが終了する

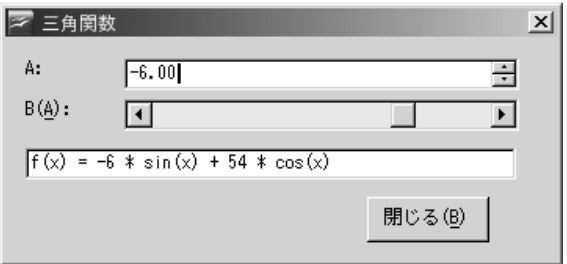
コントロールのプロパティを取得する例（コード11）

```
1: Sub ReCalculate
2:   Dim oConstANumericField As Object
3:   Dim oConstBScrollBar As Object
4:   Dim oFunctionTextField As Object
5:   Dim A As Double
6:   Dim B As Double
7:
8:   oConstANumericField = oDialog.GetControl
9:       ("ConstANumericField")
10:  oConstBScrollBar = oDialog.GetControl ("ConstBScrollBar")
11:  oFunctionTextField = oDialog.GetControl ("FunctionTextField")
12:
13:  A = oConstANumericField.Value
14:  B = oConstBScrollBar.Value
15:  oFunctionTextField.Text = "f(x) = " + A + " * sin(x) + " + B
16:  + " * cos(x) "
17: End Sub
```

マクロからダイアログボックス上に配置されたコントロールを操作するには、まずコントロールのオブジェクト参照が必要です。ダイアログオブジェクトのGetControlメソッドにダイアログエディタで設定した名前を引数として渡すと、そのコントロールのオ

プロジェクト参照が返ってきます。このオブジェクトのプロパティを設定したり、メソッドを呼び出すことにより、コントロールの操作を行ったり、あるいはコントロールのプロパティを取得することができます。Valueプロパティは、入力されている数値に対応し、Textプロパティは、入力されている文字列に対応します。

番号フィールドやスクロールバーを操作している様子



9

マクロで処理を自動化する

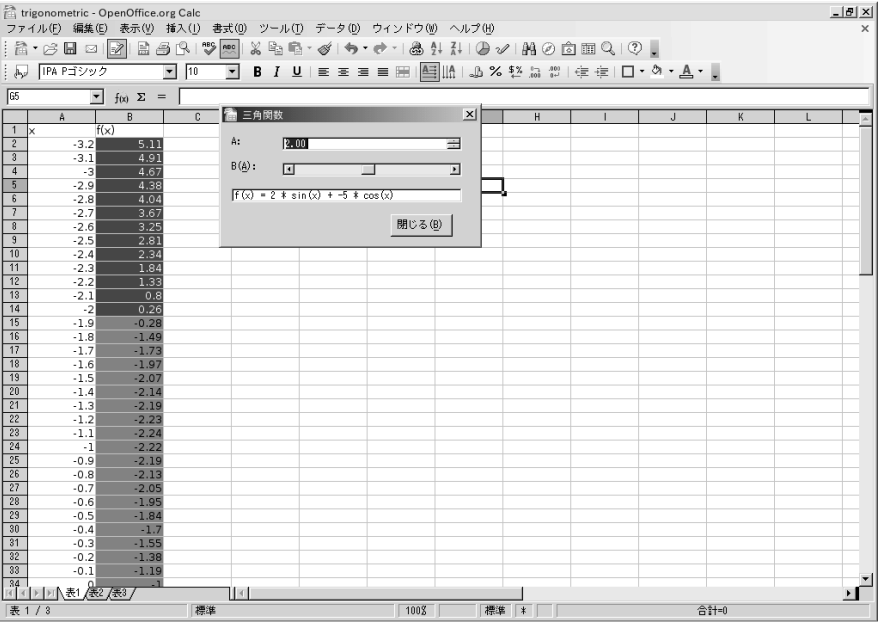
表計算ドキュメントの処理の自動化

マクロから表計算ドキュメントを操作することができます。表計算ドキュメントに限らず、文書ドキュメントなどの処理も行うことができますが、マクロがよく使われる表計算ドキュメントにおける処理をマクロにより自動化する方法を、ここでは解説します。

ここでのサンプルマクロの内容

前節に作成したダイアログを利用して、表計算ドキュメントに - 3.2 ~ 3.2までの0.1刻みのxの値に対して、ダイアログで対話式に入力した三角関数のf(x)の数式を自動的に入力するようなマクロを作ってみましょう。

ダイアログボックスに対する操作に対応して表計算ドキュメントが変更される様子



ManipulateSheet プロシージャ

番号フィールドに入力された値やスクロールバーの状態が変更されたら、A列には2行目から - 3.2 ~ 3.2の値を0.1おきに入力し、B列には2行目からf(x)の値を表示するような数式を入力するようにしましょう。

9

マクロで処理を自動化する

① コード12を「TriFunc」モジュールに追加する

表計算ドキュメントを操作するプロシージャ（コード12）

```
1: Sub ManipulateSheet (A As Double, B As Double)
2:   Dim oDocument As Object
3:   Dim oSheet As Object
4:   Dim oCell As Object
5:   Dim I As Integer
6:
7:   oDocument = ThisComponent
8:   oSheet = oDocument.Sheets(0)
9:
10:  oCell = oSheet.GetCellByPosition(0,0)
11:  oCell.String = "x"
12:
13:  oCell = oSheet.GetCellByPosition(1,0)
14:  oCell.String = "f(x)"
15:
16:  For I = 1 To 65
17:    oCell = oSheet.GetCellByPosition(0,I)
18:    oCell.Value = -3.3 + I / 10
19:
20:    oCell = oSheet.GetCellByPosition(1,I)
21:    oCell.Formula = "=" + A + "*sin(A" + (I+1) + ")+ " + B + "
      *cos(A" + (I+1) + ")"
22:
23:    If oCell.Value > 0 Then
24:      oCell.CellBackColor = RGB(0,0,255)
25:    ElseIf oCell.Value = 0 Then
26:      oCell.CellBackColor = RGB(255,255,255)
27:    Else
28:      oCell.CellBackColor = RGB(255,0,0)
29:    End If
30:  Next I
31: End Sub
```

② コード11の15行目と16行目の間に「ManipulateSheet(A,B)」という行を追加する

③ 「trigonometric.ods」を開いているCalcのウィンドウをアクティブにし、「表1」シートを表示させる

④ メニューの[ツール (T)] [マクロ (M)] [マクロを実行 (U)] を選択する

⑤ 「マクロの選択」ダイアログボックスが表示されたら「ライブラリ」から「trigonometric.ods」の左側にある ボタンをクリックする

⑥ 「Trigonometric」の左側にある ボタンをクリックし「TriFunc」をクリックする

⑦ 「マクロ名」の中から「ShowDialog」をクリックし、 ボタンをクリックする

⑧ 「TriFuncDialog」ダイアログボックスが表示される

番号フィールドの値を変更したり、スクロールバーを動かしたりすると、それに応じてCalcドキュメントのxの値に応じたf(x)の値が背景色付き（負なら赤、0なら白、正なら青）で自動的に入力されます。

ちなみに、A1からB66までの範囲を対象にしてグラフを作成すると、番号フィールドの変更やスクロールバーの動作に対応して、グラフが変化ようになります。

マクロによる表計算ドキュメントの操作の基本

サンプルマクロはうまく動作しましたか？ ここでは、コード12の中から表計算ドキュメントを操作するためのコードを解説します。表計算ドキュメントを操作するコードの基本を覚えてしまえば、あとはそれらをうまく組み合わせることにより、マクロで表計算ドキュメントの操作を自動化させたいことの大半が実現できるでしょう。

シートとセルの取得

ThisComponentはアクティブなドキュメントを指します。8行目でアクティブなドキュメントの1番目（開始値は0）のシートを取得し、oSheetに代入しています。

10行目はA1セルの参照を取得しています。シートオブジェクトのGetCellByPositionメソッドはA1セルを原点（0,0）とするXY座標を指定することでセルの参照を取得することができるメソッドです。

セルの内容の操作

セルを取得すると、セルの内容を操作することができます。11行目のようにStringプロパティに文字列を代入すると、そのセルには代入した文字列が入力されます。また、18行目のようにValueプロパティに数値を代入すると、そのセルには数値が入力されます。セルの書式設定がデフォルトのままであれば、Stringプロパティに代入された文字列は左揃え、Valueプロパティに代入された数値は右揃えで表示されます。Formulaプロパティに代入された文字列は、数式として扱われ、その数式を計算した結果がセルに表示されます。

ちなみに、String、Value、Formulaはそれぞれがプロパティなので、これらのプロパティを使ってすでにセルに入力された文字列、数値または数式が取得できます。

① 新しくCalcドキュメントを開く

② [ファイル (F)] [名前をつけて保存 (A)] でshowcontent.odsとして保存する

- ③ メニューから [ツール (T)] [マクロ (M)] [マクロの管理 (O)] [OpenOffice.org BASIC] を選択する
- ④ 「OpenOffice.org BASIC マクロ」が表示されたら [管理 (O)] ボタンをクリックする
- ⑤ 「OpenOffice.org マクロの管理」ダイアログボックスが表示されるので「モジュール」タブをクリックして表示する
- ⑥ 「モジュール」リストの中の「showcontent.ods」の左側にある ☐ マークをクリックして showcontent.ods に添付されているライブラリを表示する
- ⑦ 「Standard」を選択し、 ☐ 新規作成 (N) ボタンをクリックする
- ⑧ 「新しいモジュール」ダイアログボックスの「名前」欄には「Module1」と入力されているはずなので、そのまま ☐ OK ボタンをクリックする
- ⑨ 「Standard」ライブラリに「Module1」モジュールが追加された
- ⑩ 「Module1」を選択し、 ☐ 編集 (E) ボタンをクリックするとOpenOffice.org BASIC IDE が起動し、「Module1」モジュールを編集できるようになる
- ⑪ 「Module1」モジュールにコード13を追加する

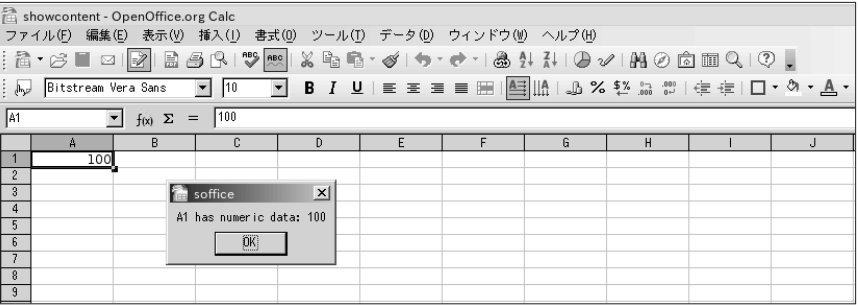
A1セルに入力されたデータをメッセージボックスに表示させるプロシージャ（コード13）

```
01 : Sub ShowA1Content
02 :   Dim Val As Double
03 :   Dim Str As String
04 :   Dim Form As String
05 :   Dim oDocument As Object
06 :   Dim oSheet As Object
07 :   Dim oCell As Object
08 :
09 :   oDocument = ThisComponent
10 :   oSheet = oDocument.Sheets(0)
11 :
12 :   oCell = oSheet.GetCellByPosition(0,0)
13 :
14 :   Select Case oCell.GetType()
15 :     Case com.sun.star.table.CellContentType.EMPTY
16 :       MsgBox "A1 is empty."
17 :     Case com.sun.star.table.CellContentType.VALUE
18 :       Val = oCell.Value
19 :       MsgBox "A1 has numeric data: " + Val
20 :     Case com.sun.star.table.CellContentType.TEXT
21 :       Str = oCell.String
22 :       MsgBox "A1 has string data: " + Str
23 :     Case com.sun.star.table.CellContentType.FORMULA
24 :       Form = oCell.Formula
25 :       MsgBox "A1 has a formula: " + Form
```

```
26 :   End Select
27 : End Sub
```

- ⑫ 「showcontent.ods」を開いているCalcのウィンドウをアクティブにし、「表1」シートを表示させる
- ⑬ A1セルに半角英数字で「100」と入力する
- ⑭ メニュー [ツール (T)] [マクロ (M)] [マクロを実行 (U)] を選択する
- ⑮ マクロセクターが表示されたら、「ライブラリ」から「showcontent.ods」の左側にある ☐ ボタンをクリックする
- ⑯ 「Standard」の左側にある + ボタンをクリックし、「Module1」をクリックする
- ⑰ 「マクロ名」から「ShowA1Content」をクリックし、 ☐ 実行 ボタンをクリックする
- ⑱ 「A1 has numeric data: 100」と書かれたメッセージボックスが表示されるので ☐ OK ボタンをクリックしてマクロを終了させる

ShowA1Contentを実行している様子



A1セルにいろいろなデータを入力して、 ~ の操作を繰り返してみてください。
GetTypeメソッドは、セルに入力されたデータの種別を定数で返します。
com.sun.star.table.CellContentType.EMPTY（空白） VALUE（数値） TEXT（文字列） FORMULA（数式）のどれかです。セルにどの種類のデータが入っているのかは分からないので、このようにしてGetTypeメソッドの返り値により処理を分けなければいけません。

セルの書式設定

コード12の24、26、28行目にあるように、CellBackColorはセルの背景色を設定するためのプロパティです。RGB関数は与えられた引数（各引数は赤、緑、青の成分を0～255で表す）から、色を表すLong型の値を返します。CellBackColorプロパティはセルの背景色を設定するためのプロパティで、RGB関数などによって生成されるLong型の値を受け付けます。

UNOを理解する

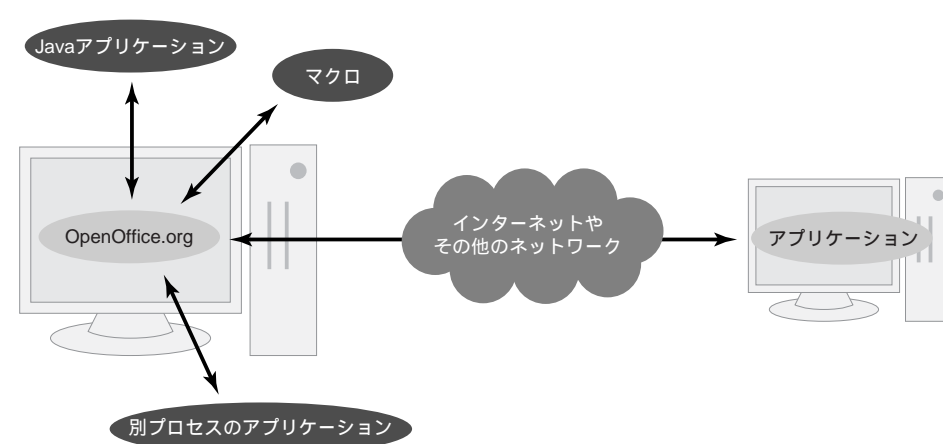
OpenOffice.orgはUNO（ユーノウ）をもとに成り立っているため、UNOのことをよく理解することはとても重要です。UNOを理解すると、OpenOffice.orgをプログラムにより操作するために参照するAPIリファレンスを見ることができるようになります。すると、マクロにより自動化するコードをどのように書けばいいのかわからない場合も、APIリファレンスを見れば分かるようになります。UNOについて学び、面倒な操作をどんどんマクロにしてしまい、もっと便利にOpenOffice.orgを使えこなせるようになります。

UNOとは

UNO（Universal Network Object）はOpenOffice.orgの基礎を成しているコンポーネント技術です。インターフェースを介して、異なるプログラミング言語、オブジェクトモデル、アーキテクチャ、プロセス、さらにはLANやインターネットをまたいで特定のオブジェクトにアクセスすることができます。

UNOはプログラミング言語としてC++、Java、Pythonなどの多数の言語をサポートしています。また、UNOコンポーネントにアクセスするだけならばOpenOffice.org BASICを使用することもできます。

UNOによるオブジェクトへのアクセス



OpenOffice.orgはこのUNOをもとに成り立っているため、UNOを紹介することにより、OpenOffice.org BASICの他、JavaやC++を使ってOpenOffice.orgのオブジェクトにアクセスする、つまりドキュメントを操作したりOpenOffice.orgの機能を利用することができます。マクロでも、このUNOをもとにドキュメントの操作を行うことになります。

サービスとIDL

UNOには特殊な概念や用語があるため、しっかりと理解しましょう。

サービスの概念

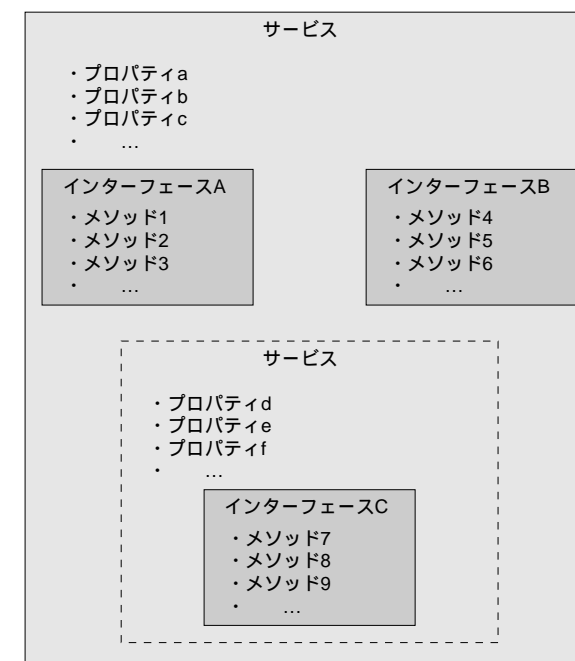
UNOにおいてサービスとは、オブジェクトの仕様のことを指します。オブジェクト指向プログラミング言語でいうところのクラスにあたります。オブジェクトは少なくとも1つ以上のサービスをサポートしなければいけません（サービスという仕様を満たすこと）。

サービスの仕様を記述するのに、IDL（Interface Definition Language）を使用しています。IDLは他のプログラムからオブジェクトを利用するために、そのオブジェクトの外部仕様を記述するための言語です。サービスは複数のインターフェースとプロパティから成っており、IDLを使ってどのインターフェースとプロパティをサポートするのかが指示されます。

ここでいうインターフェースとは、メソッドの仕様の塊のことを表します。いくつかのメソッドを機能別にまとめたものがインターフェースということになります。つまり、サービスはインターフェースがサポートするすべてのメソッドをサポートすることになります。インターフェースもIDLを使用して定義されます。インターフェースも継承することができますが、多重継承はできません。

ちなみに、サービスは他のいくつかのサービスもサポートすることができます。オブジェクト指向言語でいうところの、クラスの多重継承にあたります。

サービスの概略図



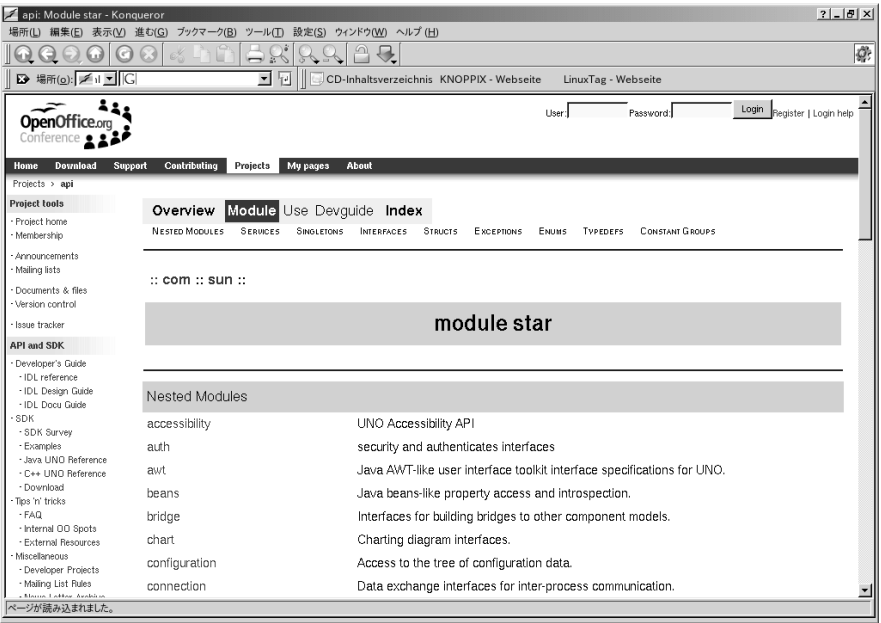
図のように、サービス が、さらにサービス をサポートしている場合、サービスをもとに作られるオブジェクトはプロパティa～f、メソッド1～9を持つことになります。

APIリファレンスの参照

OpenOffice.orgのAPIリファレンスは以下のURLにあります。

<http://api.openoffice.org/docs/common/ref/com/sun/star/module-ix.html>

APIリファレンスページ



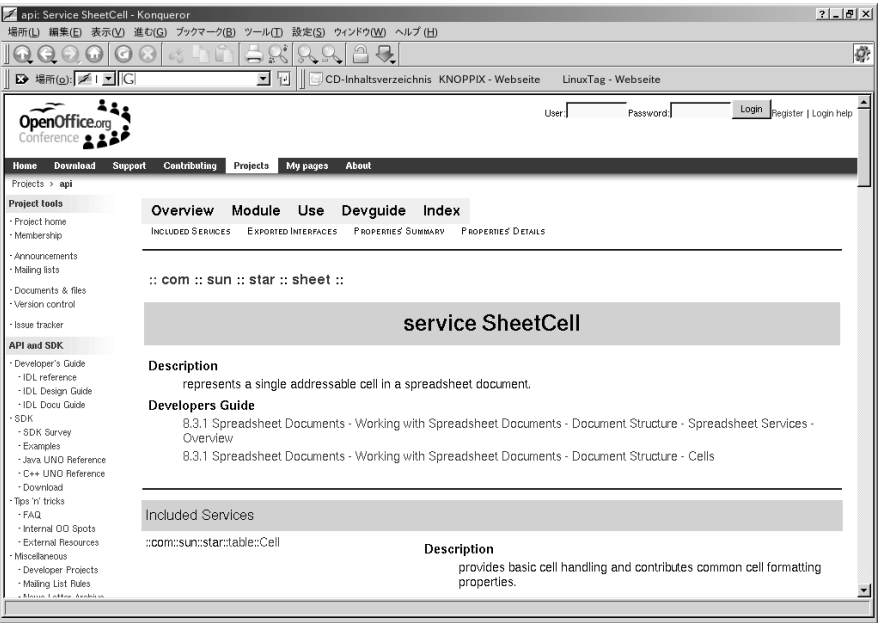
ここでは、セルオブジェクトがサポートするcom.sun.star.sheet.SheetCellサービスについて見てみましょう。

ブラウザでAPIリファレンスを開くとcom.sun.starモジュールが表示されている

sheetというリンクをクリックすると、com.sun.star.sheetモジュールに含まれているサービスやインタフェースがブラウザに表示される

SheetCellというリンクをクリックするとcom.sun.star.sheet.SheetCellサービスの詳細が表示される

SheetCellサービスのリファレンス



リファレンスを見てみると、SheetCellサービスはXActionLockableやXReplaceableインターフェースをサポートしていることが分かります（OpenOffice.org APIにおいては、インターフェース名はXで始まる）。ここでは、XReplaceableインターフェースについて見てみましょう。createReplaceDescriptorやreplaceAllというメソッドをサポートする他、XSearchableインターフェースを継承しているため、createSearchDescriptorといったメソッドもサポートすることが分かります。

また、SheetCellサービスはPositionやSizeといったプロパティをサポートします。

さらに、SheetCellサービスはCellやTextといったサービスを含んでいることが分かります。ここでは、CharacterPropertiesサービスを見てみましょう。このサービスには文字の書式設定をするためのプロパティが含まれています。たとえば、ChrColorというプロパティは文字色を設定することができるプロパティです。コード12（350ページ）の24行目を

Cell.CharColor = RGB(0,0,255)

とすると、セル内のテキストの文字色が青になります。このようにして、IDLリファレンスを見れば、オブジェクトがどのようなプロパティやメソッドを持っているのか、またその用途や引数や返り値の情報を得ることができます。

オブジェクトがサポートするサービスの見つけ方

331ページで紹介したオブザーバを使用すれば、UNOオブジェクトのプロパティなどを見ることができます。コード13を使用し、セルオブジェクトがどのサービスをサポートするのかを調べてみましょう。

- ① メニューから [ツール (T)] [マクロ (M)] [マクロの管理 (O)] [OpenOffice.org BASIC] を選択する
- ② 「OpenOffice.org BASIC マクロ」ダイアログボックスが表示されたら、「マクロの記録先」の中から「showcontents.ods」-「Standard」-「Module1」が選択されていることを確認する
- ③ をクリックする
- ④ コード13の14行目をクリックし、カーソルを14行目に置く
- ⑤ 「マクロ」ツールバーの ボタンをクリックするか キーを押すと、14行目の左側にブレークポイントを設定したことを示した赤丸のマークが表示される
- ⑥ 編集ウィンドウの中のoCellをドラッグして選択し、「マクロ」ツールバーの ボタンをクリックするか キーを押すと「オブザーバ」ウィンドウの「値」列にoCellと表示された行が追加される
- ⑦ メニューから [ツール (T)] [マクロ (M)] [マクロの管理 (O)] [OpenOffice.org BASIC] を選択する
- ⑧ 「OpenOffice.org BASICマクロ」ダイアログボックスが表示されたら「showcontent.ods」の左側にある ボタンをクリックする
- ⑨ 「Standard」の左側にある ボタンをクリックし「Module1」をクリックする
- ⑩ 「マクロのある場所」の中から「ShowA1Content」をクリックし ボタンをクリックする
- ⑪ 14行目を実行する直前で処理が一時中断する

「オブザーバ」ウィンドウに表示されたoCell変数

オブザーバ:	<input type="text" value="oCell"/>	<input type="button" value="編集"/>
変数	値	種類
<input type="checkbox"/> oCell	ScCell10bj	

「オブザーバ」ウィンドウのoCellの左側にある マークをクリックする

「オブザーバ」ウィンドウを上下にスクロールさせ、「SupportedServiceNames」という項目を探し、その左側にある マークをクリックする

SupportedServiceNames プロパティ

オブザーバ:	<input type="text" value="SupportedServiceNames"/>	<input type="button" value="編集"/>
変数	値	種類
<input type="checkbox"/> Data	Null	Object
<input type="checkbox"/> RowDescriptions		String(0 to 0)
<input type="checkbox"/> ColumnDescriptions		String(0 to 0)
<input type="checkbox"/> ImplementationName	"ScCell10bj"	String
<input checked="" type="checkbox"/> SupportedServiceNames		String(0 to 6)
Types	Null	Object
<input type="checkbox"/> ImplementationId		Integer (0 to 15)

- ⑭ oCellに格納されたオブジェクトがサポートするサービス名が列挙される

見づらい場合には、「値」行と「種類」行の間にある線を右にドラッグすると、サービス名が見やすくなります。また、「オブザーバ」ウィンドウと編集ウィンドウの間の線をドラッグすると「オブザーバ」ウィンドウの大きさを変更することができるようになります。上にドラッグすると、スクロールすることなく見られるプロパティの数が増えます。

これにより、セルを指すオブジェクトは次の7つのサービスをサポートすることが分かります。

- ・ com.sun.star.sheet.SheetCell
- ・ com.sun.star.table.Cell
- ・ com.sun.star.table.CellProperties
- ・ com.sun.star.style.CharacterProperties
- ・ com.sun.star.style.ParagraphProperties
- ・ com.sun.star.sheet.SheetCellRange
- ・ com.sun.star.table.CellRange

APIリファレンスでこれらのサービスについて調べることにより、セルを指すオブジェクトがどのようなプロパティやメソッドを持ち、それぞれどのような役割があるのかという説明を見ることができるようになります。

oCellに格納されたオブジェクトがサポートするサービス名

オブザーバ:	<input type="text" value="SupportedServiceNames"/>	<input type="button" value="編集"/>
変数	値	種類
<input type="checkbox"/> ImplementationName	"ScCell10bj"	String
<input checked="" type="checkbox"/> SupportedServiceNames		String(0 to 6)
<input type="checkbox"/> SupportedServiceNames(0)	"com.sun.star.sheet.SheetCell"	String
<input type="checkbox"/> SupportedServiceNames(1)	"com.sun.star.table.Cell"	String
<input type="checkbox"/> SupportedServiceNames(2)	"com.sun.star.table.CellProperties"	String
<input type="checkbox"/> SupportedServiceNames(3)	"com.sun.star.style.CharacterProperties"	String
<input type="checkbox"/> SupportedServiceNames(4)	"com.sun.star.style.ParagraphProperties"	String
<input type="checkbox"/> SupportedServiceNames(5)	"com.sun.star.sheet.SheetCellRange"	String
<input type="checkbox"/> SupportedServiceNames(6)	"com.sun.star.table.CellRange"	String
Types	Null	Object

このように、「オブザーバ」ウィンドウでは変数だけでなく、オブジェクトのプロパティも監視したり変更したりすることができます。

見せかけのプロパティ

APIリファレンスを見ていると、セルオブジェクトがサポートするいずれのサービスもコード12の18行目にあるようなValueというプロパティは存在しないことが分かります。代わりに、XCellというインターフェースがsetValueとgetValueというメソッドを持っていることが分かります。OpenOffice.org BASICでは、このようなメソッドをあたかもプロパティとして扱えるようになっています。

また、getByIndexというメソッドに関しては、メソッド名を省略し、配列のようにして括弧内にインデックス値を指定することもできます。

たとえば、コード12の8行目は本来ならば

```
oSheet = oDocument.getSheets().getByIndex(0)
```

と書くところを省略して書いています。

UNOオブジェクトの生成

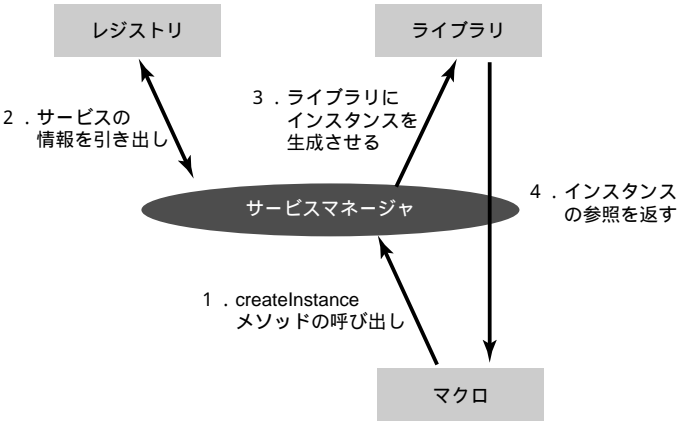
すでにあるオブジェクトにアクセスするだけでなく、オブジェクトを新たに生成することもできます。

UNOオブジェクトの生成方法

まず、はじめにUNOコンポーネントを利用するためには、サービスマネージャと呼ばれるものに使用したいサービスがあるかどうかを問い合わせます。サービスマネージャはレジストリの中から、指定されたサービスの情報を引き出し、C++やJavaで実装されたライブラリにインスタンスを生成させ、そのリファレンスを返します。

OpenOffice.org BASICでは、ランタイムがあらかじめThisComponent変数を用意していますが、これもサービスマネージャの1つです。また、サービスマネージャはcreateInstanceメソッドを持っており、このメソッドを呼び出すことにより、上述で行っている作業をコンピュータに行わせます。

UNOオブジェクトの生成方法の仕組み



新しいシートの追加

ドキュメントに新しくシートを追加するには、シートオブジェクトを作成しなければいけません。コード12の8行目を以下のように書き換えて、ShowDialogプロシージャを実行してみてください。

TriFuncというシートを得るコード（コード14）

```
1: If oDocument.Sheets.hasByName("TriFunc") Then
2:   oSheet = oDocument.Sheets.getByName("TriFunc")
3: Else
4:   oSheet = oDocument.createInstance("com.sun.star.sheet.
      Spreadsheet")
5:   oDocument.Sheets.insertByName("TriFunc", oSheet)
6: End If
```

このコードは「TriFunc」というシートがあるならばそのシートを取得します。そうでないのならば、「TriFunc」というシートを新しく作成します。

このように、新しくシートを作成するにはcreateInstanceメソッドにより「com.sun.star.sheet.SpreadSheet」サービスのインスタンスを作成します。インスタンスを作成するだけでは何も意味がないので、insertByNameメソッドを呼び出し、新しく作成したシートをドキュメントに追加します。

マクロを割り当てる

作成したマクロはツールバーやショートカットキーに割り当てることにより、より簡単に素早くマクロを呼び出せるようになります。

ツールバーのボタンに割り当てる

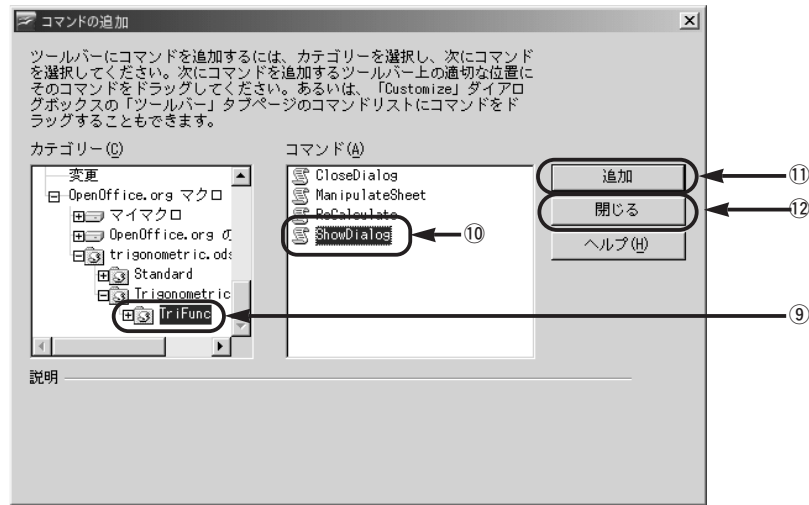
まずはツールバーにマクロを割り当ててみましょう。2節で作成したShowDialogプロシージャをツールバー上のボタンから呼び出せるようにしてみます。

- ① trigonometric.odsを開く
- ② メニューから[ツール(T)] [カスタマイズ(C)]を選択すると「カスタマイズ」ダイアログボックスが表示される
- ③ 「ツールバー」タブをクリックして表示する
- ④ 「保存場所」を「trigonometric.ods」に変更する
- ⑤ 新規(B) ボタンをクリックする



- ⑥ 「名前」ダイアログボックスが表示されるので、「ツールバー名」を「三角関数」に変更し、OK ボタンをクリックする
- ⑦ 「三角関数」というツールバーが新たに追加される
- ⑧ 追加(D) ボタンをクリックする
- ⑨ 「コマンドの追加」ダイアログが表示されるので「カテゴリー」の中から「OpenOffice.org マクロ」- 「trigonometric.ods」- 「Trigonometric」- 「TriFunc」を選択する
- ⑩ 「コマンド」の中から「ShowDialog」を選択する
- ⑪ 追加 ボタンをクリックする
- ⑫ 閉じる ボタンをクリックして「カスタマイズ」ダイアログボックスに戻る

「コマンドの追加」ダイアログボックス



- ⑬ 「コマンド」欄に「ShowDialog」が追加される

「ShowDialog」が追加されたところ



- ⑭ 変更 (M) ボタンをクリックし、[名前の変更 (A)] を選択する
- ⑮ 「名前」ダイアログボックスで「ダイアログの表示」と入力して [OK] ボタンをクリックする
- ⑯ 「ShowDialog」が「ダイアログの表示」に変更される
- ⑰ [OK] ボタンをクリックする



「ダイアログの表示」と書かれたツールバーが表示されます。このボタンをクリックすると「ShowDialog」プロシージャが実行され、TriFuncダイアログが表示されます。メニューから [ファイル (F)] [保存 (S)] を選択し、ファイルを保存するとツールバーの設定がファイルに保存されます。

「三角関数」ツールバーを追加した



このように、保存場所を特定のファイルにすると、そのファイルを開いたときにのみそのツールバーが表示されます。どのCalcドキュメントを開いたときにも表示させたい場合には、保存場所として「OpenOffice.org Calc」を選択してください。また、自分で作成した「三角関数」ツールバーを非表示にするには、[表示 (V)] [ツールバー (T)] [三角関数] を選択します。再び表示するには同じ操作を行います。

イベントに割り当てる

イベントにマクロを割り当てることにより、ファイルを開いたりするときなどに自動的にマクロを実行するようにできます。trigonometric.odsを開いたときに自動的に「ShowDialog」プロシージャが実行されるようにしてみましょう。

trigonometric.odsを開く

メニューから [ツール (T)] [カスタマイズ (C)] を選択して「カスタマイズ」ダイアログボックスを表示する
「イベント」タブをクリックして表示する
「保存場所 (A)」が「trigonometric.ods」であることを確認し、「イベント」リストから「ドキュメントを開く」を選択する
[マクロの割り当て (B)] ボタンをクリックし、「マクロの選択」ダイアログボックスを表示する



「ライブラリ (A)」から「trigonometric.ods」 - 「Trigonometric」 - 「TriFunc」を選択する

「マクロ名 (B)」から「ShowDialog」を選択し、 ボタンをクリックする

これでtrigonometric.odsを開いたときに「ShowDialog」が実行されるようになったので、 ボタンをクリックし、「カスタマイズ」ダイアログボックスを閉じる

これで、[ファイル (F)] [保存する (S)] を選択しファイルを保存すると、イベントの設定がファイルに保存されます。trigonometric.odsを閉じ、もう一度開くと自動的に「ShowDialog」プロシージャが実行されます。



このイベントの設定を解除するには、次のように操作します。

[ツール (T)] [カスタマイズ (C)] を選択して「カスタマイズ」ダイアログを表示する

「イベント」タブをクリックして表示する

「保存場所 (A)」を「trigonometric.ods」であることを確認する

「イベント」リストから「ドキュメントを開く」を選択する

をクリックする