Microsoft® Excel®

マクロ／VBA 超実践トレーニング

Office 2021／2019／2016／Microsoft 365 対応

Practice 標準解答

第1章 マクロの基本

1-1 定型作業をマクロに記録するには？

❶ セル【A1】を選択します。
❷《開発》タブを選択します。
❸《コード》グループの［マクロの記録］（マクロの記録）をクリックします。
《マクロの記録》ダイアログボックスが表示されます。
❹《マクロ名》に「文字列の太字設定」と入力します。
❺《マクロの保存先》が「作業中のブック」であることを確認します。
❻《OK》をクリックします。
マクロの記録が開始されます。
※これ以降の操作はすべて記録されます。不要な操作をしないように注意しましょう。
※マクロの記録が開始すると、［マクロの記録］（マクロの記録）が［記録終了］（記録終了）に変わります。
❼《ホーム》タブを選択します。
❽《フォント》グループの［B］（太字）をクリックします。
セル【A1】の文字列が太字になります。
❾《開発》タブを選択します。
❿《コード》グループの［記録終了］（記録終了）をクリックします。

1-2 記録したマクロを実行するには？

❶ セル範囲【A3：D3】を選択します。
❷《開発》タブを選択します。
❸《コード》グループの［マクロ］（マクロの表示）をクリックします。
《マクロ》ダイアログボックスが表示されます。
❹《マクロ名》の一覧から「文字列の太字設定」を選択します。
❺《実行》をクリックします。
マクロが実行され、セル範囲【A3：D3】の文字列が太字になります。

1-3 マクロを有効化して開いたり保存したりするには？

まずは、マクロを有効化します。

❶ ブック「1-3Practice」を開きます。

❷ メッセージバーにセキュリティの警告が表示されていることを確認します。

❸《コンテンツの有効化》をクリックします。

マクロが有効になります。

続いて、マクロ有効ブックとしてブックを保存します。

❹《ファイル》タブを選択します。

❺《エクスポート》をクリックします。

❻《ファイルの種類の変更》をクリックします。

❼ 右側の一覧から《マクロ有効ブック》を選択します。

❽《名前を付けて保存》をクリックします。

《名前を付けて保存》ダイアログボックスが表示されます。

❾ フォルダー「第1章」を選択します。

❿《ファイル名》に「マクロ登録練習」と入力します。

⓫《ファイルの種類》が《Excelマクロ有効ブック》になっていることを確認します。

⓬《保存》をクリックします。

ブックが保存されます。

1-4 マクロを指定のブックに保存するには？

❶《開発》タブを選択します。
❷《コード》グループの[マクロの記録]（マクロの記録）をクリックします。
《マクロの記録》ダイアログボックスが表示されます。
❸《マクロ名》に「フォントの設定」と入力します。
❹《マクロの保存先》の[⌄]をクリックし、一覧から「新しいブック」を選択します。
❺《OK》をクリックします。
新しいブックが開きます。
❻ ブック「1-4Practice」をアクティブにします。
❼ Ctrl＋Aを押します。
シートの全範囲が選択されます。
❽《ホーム》タブを選択します。
❾《フォント》グループの[游ゴシック ⌄]（フォント）の[⌄]をクリックします。
❿ 一覧から「メイリオ」を選択します。
シート内の文字列のフォントが「メイリオ」に設定されます。
⓫《開発》タブを選択します。
⓬《コード》グループの[記録終了]（記録終了）をクリックします。
⓭ 新しいブックをアクティブにします。
⓮《ファイル》タブを選択します。
⓯《エクスポート》をクリックします。
⓰《ファイルの種類の変更》をクリックします。
⓱ 右側の一覧から《マクロ有効ブック》を選択します。
⓲《名前を付けて保存》をクリックします。
《名前を付けて保存》ダイアログボックスが表示されます。
⓳ 保存先のフォルダーに「第1章」を選択します。
⓴《ファイル名》に「マクロ登録用」と入力します。
㉑《ファイルの種類》が《Excelマクロ有効ブック》になっていることを確認します。
㉒《保存》をクリックします。

1-5 マクロの不要な行を削除・コピーするには？

※VBEを起動し、《標準モジュール》→「Module1」を開いておきましょう。

まずは、マクロから不要な行を削除します。

❶ コードウィンドウにカーソルを移動し、次の行を選択し、削除します。

```
.Strikethrough = False
.Superscript = False
.Subscript = False
.OutlineFont = False
.Shadow = False
.Underline = xlUnderlineStyleNone
.ThemeColor = xlThemeColorLight1
.TintAndShade = 0
.ThemeFont = xlThemeFontNone
```

❷「'」で始まるコメント行と空白行（2～6行目）を選択し、削除します。

続いて、マクロをコピーしてマクロ名を変更します。

❸ マクロ「**フォントの設定**」のすべての行を選択します。

❹  （コピー）をクリックします。

❺ 最終行の下へカーソルを移動します。

❻  （貼り付け）をクリックします。

❼ 貼り付けたコードのマクロ名を「**フォントの設定2**」に変更します。

※上書き保存しておきましょう。

1-6 マクロを編集しコンパイルを実行するには？

※VBEを起動し、《標準モジュール》→「Module1」を開いておきましょう。

❶ マクロ名「**フォントの設定2**」を「**明朝体に設定**」に変更します。

❷ マクロ「**明朝体に設定**」のコードを次のように変更します。

```
.Name = "MS　明朝"
```

❸《デバッグ》をクリックします。

❹《VBAProjectのコンパイル》をクリックします。

※コードの文法に間違いがない場合は、何も表示されません。上書き保存しておきましょう。

第2章 モジュールとプロシージャの基本

2-1 モジュールを作成するには？

※VBEを起動しておきましょう。

まずは、モジュールの削除を行います。

❶ プロジェクトエクスプローラーのモジュール「keisan」を右クリックします。

❷《keisanの解放》をクリックします。

「削除する前にkeisanをエクスポートしますか？」とメッセージが表示されます。

❸《いいえ》をクリックします。

モジュール「keisan」が削除されます。

続いて、新しいモジュールを作成します。

❹《挿入》をクリックします。

❺《標準モジュール》をクリックします。

❻ プロジェクトエクスプローラーのモジュール「Module1」を選択します。

❼ プロパティウィンドウの《全体》タブを選択します。

❽《(オブジェクト名)》に「uriage」と入力します。

❾ Enter を押します。

プロジェクトエクスプローラーのモジュール名が変更されます。

2-2 プロシージャを作成するには？

※VBEを起動し、《標準モジュール》→「uriage」を開いておきましょう。

❶ コードウィンドウにカーソルを移動し、「sub 合計売上額の削除」と入力します。

❷ Enter を押します。

「End Sub」が自動的に入力され、プロシージャが作成されます。

❸ Tab を押して字下げします。

❹ 次のように入力します。

```
Range("D14").ClearContents
```

※コンパイルを実行し、上書き保存しておきましょう。

2-3 プロシージャをショートカットキーに登録するには？

❶《開発》タブを選択します。
❷《コード》グループの  （マクロの表示）をクリックします。
《マクロ》ダイアログボックスが表示されます。
❸《マクロ名》の一覧から、「フォントの設定」を選択します。
❹《オプション》をクリックします。
《マクロオプション》ダイアログボックスが表示されます。
❺《ショートカットキー》に「f」と入力します。
❻《OK》をクリックします。
《マクロ》ダイアログボックスに戻ります。
❼《キャンセル》をクリックします。
※《実行》をクリックすると、プロシージャが実行されるので注意しましょう。
※設定したショートカットキーからプロシージャが実行されるか確認しておきましょう。例えばセル範囲【A4：A12】を選択した状態でショートカットキーを実行すると、この範囲の文字列にフォントの設定が適用されます。

2-4 プロシージャをボタンや図形に登録するには？

まずは、ボタンにプロシージャを登録します。
❶《開発》タブを選択します。
❷《コントロール》グループの  （コントロールの挿入）をクリックします。
❸《フォームコントロール》の  （ボタン（フォームコントロール））をクリックします。
マウスポインターの形が＋に変わります。
❹任意の場所でドラッグします。
《マクロの登録》ダイアログボックスが表示されます。
❺《マクロ名》の一覧から「合計売上額計算」を選択します。
❻《OK》をクリックします。
❼ボタンが選択されていることを確認します。
❽「合計売上額の計算」と入力します。
※文字列を入力した後に Enter を押すと改行されるので注意しましょう。
続いて、図形にプロシージャを登録します。
❾「売上額並べ替え」の図形を右クリックします。
❿《マクロの登録》を選択します。
《マクロの登録》ダイアログボックスが表示されます。
⓫《マクロ名》の一覧から「売上額並べ替え」を選択します。
⓬《OK》をクリックします。
※任意のセルをクリックして、図形の選択を解除しておきましょう。
※ボタンや図形をクリックして、プロシージャが実行されることを確認しておきましょう。

2-5 モジュールをインポート・エクスポートするには？

※VBEを起動しておきましょう。

まずは、モジュールのインポートを行います。

❶《ファイル》タブを選択します。

❷《ファイルのインポート》をクリックします。

《ファイルのインポート》ダイアログボックスが表示されます。

❸ フォルダー「第2章」から「ソート.bas」を選択します。

❹《開く》をクリックします。

プロジェクトエクスプローラーの《標準モジュール》に、モジュール「ソート」が追加されます。

続いて、モジュールのエクスポートを行います。

❺ プロジェクトエクスプローラーから「uriage」を選択します。

❻《ファイル》タブを選択します。

❼《ファイルのエクスポート》をクリックします。

《ファイルのエクスポート》ダイアログボックスが表示されます。

❽ フォルダー「第2章」を選択します。

❾ ファイル名に「売上額」と入力します。

❿《保存》をクリックします。

第3章 オブジェクトの利用

3-1 セルのフォントを変更するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「フォントサイズの変更」プロシージャ

```
1. Sub フォントサイズの変更()
2.       Range("A1","G1").Font.Size = 14
3. End Sub
```

■プロシージャの意味

1.「フォントサイズの変更」プロシージャ開始
2.　　セル【A1】と【G1】のフォントサイズを「14」に設定
3. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

3-2 セルまたはセル範囲を選択するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「表データの選択」プロシージャ

```
1. Sub 表データの選択()
2.       Range("A4:D18").Select
3. End Sub
```

■プロシージャの意味

1.「表データの選択」プロシージャ開始
2.　　セル範囲【A4:D18】を選択
3. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

3-3 複数のプロパティを同時に設定するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「表タイトルのフォント設定」プロシージャ

```
1. Sub 表タイトルのフォント設定()
2.     With Range("A1", "G1").Font
3.         .Name = "MS ゴシック"
4.         .Size = 14
5.         .ThemeColor = 5
6.     End With
7. End Sub
```

■プロシージャの意味

```
1.「表タイトルのフォント設定」プロシージャ開始
2.     セル【A1】、【G1】のフォントを次のように設定
3.         フォント名は「MSゴシック」
4.         フォントサイズを「14」
5.         フォントのテーマの色は「アクセント1」
6.     フォントの設定を終了
7. プロシージャ終了
```

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

3-4 選択しているセル範囲に罫線を引くには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「選択範囲に二重罫線を引く」プロシージャ

```
1. Sub 選択範囲に二重罫線を引く()
2.     Selection.BorderAround xlDouble, , , vbBlue
3. End Sub
```

■プロシージャの意味

```
1.「選択範囲に二重罫線を引く」プロシージャ開始
2.     現在選択しているセル範囲の周囲に青色の二重線を引く
3. プロシージャ終了
```

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

3-5 表のひとつ下のセルを選択するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「表の右端の1列右を選択」プロシージャ

```
1. Sub 表の右端の1列右を選択()
2.     Range("A3").Select
3.     Selection.End(xlToRight).Select
4.     ActiveCell.Offset(0, 1).Select
5. End Sub
```

■プロシージャの意味

1.「表の右端の1列右を選択」プロシージャ開始
2. セル【A3】を選択
3. Ctrl ＋ → でデータの右端のセルを選択
4. アクティブセルの1列右のセルを選択
5. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

3-6 連続するセル範囲に背景色を設定するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「表の連続するセルに水色の背景色を設定」プロシージャ

```
1. Sub 表の連続するセルに水色の背景色を設定()
2.     Range("G3").Select
3.     ActiveCell.CurrentRegion.Select
4.     Selection.Interior.Color = RGB(164, 222, 228)
5. End Sub
```

■プロシージャの意味

1.「表の連続するセルに水色の背景色を設定」プロシージャ開始
2. セル【G3】を選択
3. アクティブセルから上下左右に連続する範囲を選択
4. 選択しているセルの背景色を水色に設定
5. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

3-7 行高や列幅を自動調整するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「行の高さの自動調整と列の非表示」プロシージャ

```
1. Sub 行の高さの自動調整と列の非表示()
2.     Rows(3).AutoFit
3.     Columns("G").Hidden = True
4. End Sub
```

■プロシージャの意味

1.「行の高さの自動調整と列の非表示」プロシージャ開始
2.　　3行目の行の高さを自動調整
3.　　G列を非表示にする
4. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

3-8 セルを指定の形式でコピーするには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「書式のみコピー」プロシージャ

```
1. Sub 書式のみコピー()
2.     Range("A3").Copy
3.     Range("E3").PasteSpecial Paste:=xlPasteFormats
4.     Application.CutCopyMode = False
5. End Sub
```

■プロシージャの意味

1.「書式のみコピー」プロシージャ開始
2.　　セル【A3】をコピー
3.　　セル【E3】に書式だけを貼り付ける
4.　　コピーモードを解除
5. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

3-9 選択しているセルやセル範囲に名前を付けるには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「換算表に名前を設定」プロシージャ

```
1. Sub 換算表に名前を設定()
2.     Range("E5:F10").Name = "換算表"
3. End Sub
```

■プロシージャの意味

1.「換算表に名前を設定」プロシージャ開始
2.　　セル範囲【E5:F10】に「換算表」と名前を設定
3. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

3-10 フィルターでデータを抽出するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「学年フィルター」プロシージャ

```
1. Sub 学年フィルター()
2.     Range("A3:G23").AutoFilter Field:=4, Criteria1:="3"
3. End Sub
```

■「学年フィルター」プロシージャの意味

1.「学年フィルター」プロシージャ開始
2.　　セル範囲【A3:G23】のうち4列目が「3」のデータを抽出
3. プロシージャ終了

■「学年フィルターの解除」プロシージャ

```
1. Sub 学年フィルターの解除()
2.     Range("A3:G23").AutoFilter
3. End Sub
```

■「学年フィルターの解除」プロシージャの意味

1.「学年フィルターの解除」プロシージャ開始
2.　　セル範囲【A3:G23】のフィルター解除
3. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

3-11 表のデータを並べ替えるには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「生徒番号降順」プロシージャ

```
1. Sub 生徒番号降順()
2.     Range("A3").Sort Key1:=Range("A3"), Order1:=xlDescending, Header:=xlYes
3. End Sub
```

■「生徒番号降順」プロシージャの意味

1.「生徒番号降順」プロシージャ開始
2. セル【A3】を含む連続するセル範囲に対して並べ替え（並べ替えフィールドはセル【A3】、降順、先頭行を見出しとする）
3. プロシージャ終了

■「生徒番号昇順」プロシージャ

```
1. Sub 生徒番号昇順()
2.     Range("A3").Sort Key1:=Range("A3"), Order1:=xlAscending, Header:=xlYes
3. End Sub
```

■「生徒番号昇順」プロシージャの意味

1.「生徒番号昇順」プロシージャ開始
2. セル【A3】を含む連続するセル範囲に対して並べ替え（並べ替えフィールドはセル【A3】、昇順、先頭行を見出しとする）
3. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

3-12 図形やグラフの表示／非表示を切り替えるには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「図形非表示」プロシージャ

```
1. Sub 図形非表示()
2.     Worksheets("案内状").Shapes("路線").Visible = False
3. End Sub
```

■「図形非表示」プロシージャの意味

1.「図形非表示」プロシージャ開始
2.　　シート「案内状」の図形「路線」を非表示にする
3. プロシージャ終了

■「図形再表示」プロシージャ

```
1. Sub 図形再表示()
2.     Worksheets("案内状").Shapes("路線").Visible = True
3. End Sub
```

■「図形再表示」プロシージャの意味

1.「図形再表示」プロシージャ開始
2.　　シート「案内状」の図形「路線」を再表示する
3. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

3-13 ワークシートを追加したり削除したりするには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「ワークシートの追加と削除」プロシージャ

```
1. Sub ワークシートの追加と削除()
2.     Worksheets.Add
3.     Worksheets("2020").Delete
4. End Sub
```

■プロシージャの意味

```
1.「ワークシートの追加と削除」プロシージャ開始
2.     ワークシートを追加
3.     ワークシート「2020」を削除
4. プロシージャ終了
```

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

3-14 ワークシートをコピーしたり移動したりするには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「ワークシートのコピーとリネーム」プロシージャ

```
1. Sub ワークシートのコピーとリネーム()
2.     Worksheets("2023").Copy After:=ActiveSheet
3.     ActiveSheet.Name = "2024"
4. End Sub
```

■プロシージャの意味

```
1.「ワークシートのコピーとリネーム」プロシージャ開始
2.     ワークシート「2023」をアクティブシートの右側（直後）にコピー
3.     アクティブシートの名前を「2024」に変更
4. プロシージャ終了
```

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

3-15 印刷のページレイアウトを設定するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「縦向き150垂直中央のページレイアウト設定」プロシージャ

```
1. Sub 縦向き150垂直中央のページレイアウト設定()
2.     With ActiveSheet.PageSetup
3.         .Orientation = xlPortrait
4.         .Zoom = 150
5.         .CenterVertically = True
6.     End With
7.     ActiveSheet.PrintPreview
8. End Sub
```

■プロシージャの意味

```
1.「縦向き150垂直中央のページレイアウト設定」プロシージャ開始
2.     アクティブシートのページレイアウトを次のように設定
3.         印刷の向きを縦向きにする
4.         拡大縮小率を150に設定する
5.         垂直方向の中央に印刷する
6.     ページレイアウトの設定を終了
7.     アクティブシートの印刷プレビューを表示
8. プロシージャ終了
```

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

3-16 ワークシートの印刷範囲を設定するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「期間スケジュールの印刷設定」プロシージャ

```
1. Sub 期間スケジュールの印刷設定()
2.     With Worksheets("スケジュール").PageSetup
3.         .PrintArea = "A1:F67"
4.         .PrintTitleRows = Rows("1:5").Address
5.     End With
6. End Sub
```

■プロシージャの意味

```
1.「期間スケジュールの印刷設定」プロシージャ開始
2.     ワークシート「スケジュール」のページレイアウトを次のように設定
3.         印刷範囲をセル範囲【A1:F67】にする
4.         タイトル行を1～5行目にする
5.     ページレイアウトの設定を終了
6. プロシージャ終了
```

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。印刷プレビューで、設定した内容が反映されているか確認しましょう。

3-17 ワークシートに改ページを追加するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「業務月別の改ページ設定」プロシージャ

```
1. Sub 業務月別の改ページ設定()
2.     With Worksheets("スケジュール")
3.         .HPageBreaks.Add Before:=Range("A37")
4.         .HPageBreaks.Add Before:=Range("A68")
5.         .PageSetup.PrintTitleRows = Rows("4:5").Address
6.     End With
7. End Sub
```

■プロシージャの意味

1.「業務月別の改ページ設定」プロシージャ開始
2.　　ワークシート「スケジュール」を次のように設定
3.　　セル【A37】の上に水平改ページを追加（8月の1行目）
4.　　セル【A68】の上に水平改ページを追加（9月の1行目）
5.　　ページ設定のタイトル行を4～5行目に設定
6.　　設定を終了
7. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。印刷プレビューで、設定した内容が反映されているか確認しましょう。

3-18 ブックを開くには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「模擬試験成績順を開く」プロシージャ

```
1. Sub 模擬試験成績順を開く()
2.     Workbooks.Open Filename:=ThisWorkbook.Path & "¥模擬試験成績順.xlsx"
3. End Sub
```

■プロシージャの意味

1.「模擬試験成績順を開く」プロシージャ開始
2.　　実行中のプロシージャが記述されたブックと同じフォルダー内のブック「模擬試験成績順.xlsx」を開く
3. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

3-19 選択したブックのパスを調べるには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「マクロファイルの絶対パスの表示」プロシージャ

```
1. Sub マクロファイルの絶対パスの表示()
2.     Dim Fname As String
3.     Fname = Application.GetOpenFilename("マクロファイル,*.xlsm")
4.     If Fname <> "False" Then
5.         Workbooks.Open Filename:=Fname
6.     End If
7.     MsgBox Fname
8. End Sub
```

■プロシージャの意味

1.「マクロファイルの絶対パスの表示」プロシージャ開始
2. 文字列型の変数「Fname」を使用することを宣言
3. 表示するファイルの種類をマクロファイルに限定して、《ファイルを開く》ダイアログボックスを表示する絶対パスを変数「Fname」に代入
4. 変数「Fname」がFalseでない場合（ファイルが選択されている場合）は
5. 変数「Fname」に代入されたブックを開く
6. Ifステートメント終了
7. 変数「Fname」の内容（選択したファイルの絶対パス）をメッセージに表示
8. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

3-20 ブックを保存するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「成績表の新規保存」プロシージャ

```
1. Sub 成績表の新規保存()
2.     ThisWorkbook.SaveAs Filename:=ThisWorkbook.Path & "¥2024模擬試験成績表.xlsm"
3. End Sub
```

■「成績表の新規保存」プロシージャの意味

1.「成績表の新規保存」プロシージャ開始
2.　　実行中のプロシージャが記述されたブックを、同じフォルダー内に「2024模擬試験成績表.xlsm」という名前で保存
3. プロシージャ終了

■「成績表の上書き保存」プロシージャ

```
1. Sub 成績表の上書き保存()
2.     ThisWorkbook.Save
3. End Sub
```

■「成績表の上書き保存」プロシージャの意味

1.「成績表の上書き保存」プロシージャ開始
2.　　実行中のブックを上書き保存
3. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

3-21 ブックを閉じるには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「ブックを保存せずに閉じる」プロシージャ

```
1. Sub ブックを保存せずに閉じる()
2.     ThisWorkbook.Close SaveChanges:=False
3. End Sub
```

■プロシージャの意味

1.「ブックを保存せずに閉じる」プロシージャ開始
2.　　変更を保存せずに実行中のブックを閉じる
3. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

第4章 変数と制御構文

4-1 変数を利用して計算を行うには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「週次計算」プロシージャ

```
1. Sub 週次計算()
2.     Dim jikan As Double
3.     Dim hoshu As Long
4.     jikan = Range("B4").Value + Range("B5").Value
5.     hoshu = Range("C4").Value + Range("C5").Value
6.     Range("B6").Value = jikan
7.     Range("C6").Value = hoshu
8. End Sub
```

■プロシージャの意味

1.「週次計算」プロシージャ開始
2. 倍精度浮動小数点数型の変数「jikan」を使用することを宣言
3. 長整数型の変数「hoshu」を使用することを宣言
4. 変数「jikan」にセル【B4】＋セル【B5】の計算結果を代入
5. 変数「hoshu」にセル【C4】＋セル【C5】の計算結果を代入
6. セル【B6】に変数「jikan」の値を設定
7. セル【C6】に変数「hoshu」の値を設定
8. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

4-2 定数を利用して計算を行うには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「月次計算」プロシージャ

```
1. Sub 月次計算()
2.     Dim jikan As Double
3.     Dim hoshu As Long
4.     Const jikyu As Integer = 1200
5.     jikan = Range("B22").Value
6.     hoshu = jikan * jikyu
7.     Range("C22").Value = hoshu
8. End Sub
```

■プロシージャの意味

1.「月次計算」プロシージャ開始
2.　倍精度浮動小数点数型の変数「jikan」を使用することを宣言
3.　長整数型の変数「hoshu」を使用することを宣言
4.　整数型の定数「jikyu」を使用することを宣言し、数値「1200」の値を指定
5.　変数「jikan」にセル【B22】の値を代入
6.　変数「hoshu」に変数「jikan」×定数「jikyu」の計算結果を代入
7.　セル【C22】に変数「hoshu」の値を設定
8. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

4-3 条件が成立した場合の処理を指定するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「診断結果判定」プロシージャ

```
1. Sub 診断結果判定()
2.     Dim hantei As Byte
3.     Const kijun As Byte = 6
4.     hantei = Range("C12").Value
5.     If hantei < kijun Then
6.         MsgBox "改善が必要です"
7.     End If
8. End Sub
```

■プロシージャの意味

```
1.「診断結果判定」プロシージャ開始
2.     バイト型の変数「hantei」を使用することを宣言
3.     バイト型の定数「kijun」を使用することを宣言し、数値「6」の値を指定
4.     変数「hantei」にセル【C12】の値を代入
5.     変数「hantei」が定数「kijun」より小さい場合は
6.         「改善が必要です」のメッセージを表示
7.     Ifステートメント終了
8. プロシージャ終了
```

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

4-4 条件の成立・不成立に応じて処理を分岐するには？

❶ 次のようにプロシージャを入力します。

※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「診断結果判定」プロシージャ

```
Sub 診断結果判定()
    Dim hantei As Byte
    Const kijun As Byte = 6
    hantei = Range("C12").Value
    If hantei > kijun Then
        MsgBox "健康です"
    Else
        MsgBox "改善が必要です"
    End If
End Sub
```

■プロシージャの意味

```
「診断結果判定」プロシージャ開始
    バイト型の変数「hantei」を使用することを宣言
    バイト型の定数「kijun」を使用することを宣言し、数値「6」の値を指定
    変数「hantei」にセル【C12】の値を代入
    変数「hantei」が定数「kijun」より大きい場合は
        「健康です」のメッセージを表示
    それ以外の場合は
        「改善が必要です」のメッセージを表示
    Ifステートメント終了
プロシージャ終了
```

※コンパイルを実行し、上書き保存しておきましょう。

※プロシージャの動作を確認します。

4-5 条件が複数ある場合の処理を指定するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「診断結果判定」プロシージャ

```
1. Sub 診断結果判定()
2.     Dim hantei As Byte
3.     Const kijun As Byte = 6
4.     Const kijun2 As Byte = 2
5.     hantei = Range("C12").Value
6.     If hantei > kijun Then
7.         MsgBox "健康です"
8.     ElseIf hantei > kijun2 Then
9.         MsgBox "改善が必要です"
10.    Else
11.        MsgBox "再検査が必要です"
12.    End If
13. End Sub
```

■プロシージャの意味

```
1.「診断結果判定」プロシージャ開始
2.     バイト型の変数「hantei」を使用することを宣言
3.     バイト型の定数「kijun」を使用することを宣言し、数値「6」の値を指定
4.     バイト型の定数「kijun2」を使用することを宣言し、数値「2」の値を指定
5.     変数「hantei」にセル【C12】の値を代入
6.     変数「hantei」が定数「kijun」より大きい場合は
7.         「健康です」のメッセージを表示
8.     変数「hantei」が定数「kijun2」より大きい場合は
9.         「改善が必要です」のメッセージを表示
10.    それ以外の場合は
11.        「再検査が必要です」のメッセージを表示
12.    Ifステートメント終了
13. プロシージャ終了
```

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

4-6 条件が多い場合に処理を分岐するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「料金種別表示」プロシージャ

```
Sub 料金種別表示()
    Dim syubetsu As String
    Select Case Range("F3").Value
        Case 0 To 6
            syubetsu = Range("A4").Value
        Case 7 To 12
            syubetsu = Range("A5").Value
        Case 13 To 15
            syubetsu = Range("A6").Value
        Case Is >= 16
            syubetsu = Range("A7").Value
        Case Else
            syubetsu = "設定されていない種別"
    End Select
    MsgBox syubetsu & "の料金です"
End Sub
```

■プロシージャの意味

```
「料金種別表示」プロシージャ開始
    文字列型の変数「syubetsu」を使用することを宣言
    セル【F3】の値が
        「0」～「6」の場合は
            変数「syubetsu」にセル【A4】の値を代入
        「7」～「12」の場合は
            変数「syubetsu」にセル【A5】の値を代入
        「13」～「15」の場合は
            変数「syubetsu」にセル【A6】の値を代入
        「16」以上の場合は
            変数「syubetsu」にセル【A7】の値を代入
        それ以外の場合は
            変数「syubetsu」に「設定されていない種別」を代入
    Select Caseステートメント終了
    変数「syubetsu」の値と他の文字列を連結してメッセージを表示
プロシージャ終了
```

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

4-7 指定した回数だけ処理を繰り返すには？

❶ 次のようにプロシージャを入力します。

※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「顧客番号付番」プロシージャ

```
1. Sub 顧客番号付番()
2.     Dim i As Integer
3.     Dim kokyakubangou As String
4.     For i = 1 To 15
5.         kokyakubangou = "5" & Format(i, "0000")
6.         Range("A3").Offset(i, 0).Value = kokyakubangou
7.     Next
8. End Sub
```

■プロシージャの意味

1.「顧客番号付番」プロシージャ開始
2. 整数型の変数「i」を使用することを宣言
3. 文字列型の変数「kokyakubangou」を使用することを宣言
4. 変数「i」が「1」から「15」になるまで次の行以降の処理を繰り返す
5. 変数「kokyakubangou」に、「5」と変数「i」を表示形式「0000」に変換した値を連結して代入
6. セル【A3】から変数「i」の行数分下に移動したセルに、変数「kokyakubangou」の値を設定
7. 変数「i」に変数「i」＋「1」の結果を代入し、4行目に戻る
8. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。

※プロシージャの動作を確認します。

4-8 コレクション内に対して処理を繰り返すには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「割引後価格の計算」プロシージャ

```
1. Sub 割引後価格の計算()
2.     Dim waribikiritu As Double
3.     Dim kakaku As Range
4.     waribikiritu = Range("D2").Value
5.     For Each kakaku In Range("D4:D13")
6.         kakaku.Value = kakaku.Offset(0, -1) * (1 - waribikiritu)
7.     Next kakaku
8. End Sub
```

■プロシージャの意味

1.「割引後価格の計算」プロシージャ開始
2. 倍精度浮動小数点数型の変数「waribikiritu」を使用することを宣言
3. Range型のオブジェクト変数「kakaku」を使用することを宣言
4. 変数「waribikiritu」にセル【D2】の値を代入
5. セル範囲【D4:D13】のすべてのセルに対して処理を繰り返す
6. オブジェクト変数「kakaku」が参照するセルに、1列左に移動したセルの値×(1-変数「waribikiritu」)の計算結果を代入
7. オブジェクト変数「kakaku」に次のセルへの参照を代入し、5行目に戻る
8. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

4-9 条件が成立している間、処理を繰り返すには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「本店文字色の変更」プロシージャ

```
1. Sub 本店文字色の変更()
2.     Dim i As Integer
3.     i = 0
4.     Do
5.         Range("A4").Offset(i, 2).Font.Color = vbRed
6.         i = i + 1
7.     Loop While Range("A4").Offset(i, 0).Value <> ""
8. End Sub
```

■プロシージャの意味

1.「本店文字色の変更」プロシージャ開始
2.　　整数型の変数「i」を使用することを宣言
3.　　変数「i」に「0」を代入
4.　　次の行以降の処理を繰り返す
5.　　セル【A4】から変数「i」の行数分下に、2列右に移動したセルのフォントの色を赤色に設定
6.　　変数「i」に変数「i」＋1の結果を代入
7.　　セル【A4】から変数「i」の行数分下に移動したセルの値が空白でない間は、5行目に戻る
8. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

第5章 関数の利用

5-1 メッセージボックスを表示するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「メッセージボックスの表示選択」プロシージャ

```
1. Sub メッセージボックスの表示選択()
2.     Dim Mymsg As Byte
3.     Mymsg = MsgBox("どちらかを選択してください。", vbYesNo, "選択画面")
4.     If Mymsg = 6 Then
5.         MsgBox "《はい》が選択されました。"
6.     ElseIf Mymsg = 7 Then
7.         MsgBox "《いいえ》が選択されました。"
8.     End If
9. End Sub
```

■プロシージャの意味

1.「メッセージボックスの表示選択」プロシージャ開始
2. バイト型の変数「Mymsg」を使用することを宣言
3. タイトルバーに「選択画面」、「はい」「いいえ」ボタンを持つメッセージボックスに「どちらかを選択してください。」と表示し、ユーザーの選択結果を変数「Mymsg」に代入
4. 変数「Mymsg」の値が「6」の場合は
5. 「《はい》が選択されました。」のメッセージを表示
6. 変数「Mymsg」の値が「7」の場合は
7. 「《いいえ》が選択されました。」のメッセージを表示
8. Ifステートメント終了
9. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

5-2 現在の日付や時刻を表示するには？

❶ 次のようにプロシージャを入力します。

※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「現在の日付と時刻の表示」プロシージャ

```
1. Sub 現在の日付と時刻の表示()
2.     MsgBox "現在の日付と時刻は、" & Now & "です。"
3. End Sub
```

■プロシージャの意味

1.「現在の日付と時刻の表示」プロシージャ開始
2.　　現在の日付と時刻を他の文字列と連結してメッセージを表示
3. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。

※プロシージャの動作を確認します。

5-3 特定の文字列を別の文字列に置換するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「講師名の変更」プロシージャ

```
1. Sub 講師名の変更()
2.     Dim Myrange As Range
3.     For Each Myrange In Range("D3:D22")
4.         Myrange.Value = Replace(Myrange.Value, "Michael Brown", "Sophia Clark")
5.     Next Myrange
6. End Sub
```

■プロシージャの意味

```
1.「講師名の変更」プロシージャ開始
2.     Range型のオブジェクト変数「Myrange」を使用することを宣言
3.     セル範囲【D3:D22】のすべてのセルに対して処理を繰り返す
4.         オブジェクト変数「Myrange」が参照するセルに、セルの文字列内の「Michael Brown」を
           「Sophia Clark」に置換した文字列を代入
5.     オブジェクト変数「Myrange」に次のセルへの参照を代入し、3行目に戻る
6. プロシージャ終了
```

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

5-4 文字列から一部を取り出すには？

❶ 次のようにプロシージャを入力します。

※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「番号の分割入力」プロシージャ

```
1. Sub 番号の分割入力()
2.     Dim Myrange As Range
3.     For Each Myrange In Range("A3:A22")
4.         Myrange.Offset(0, 5).Value = Left(Myrange.Value, 4)
5.         Myrange.Offset(0, 6).Value = Mid(Myrange.Value, 6, 1)
6.         Myrange.Offset(0, 7).Value = Right(Myrange.Value, 2)
7.     Next Myrange
8. End Sub
```

■プロシージャの意味

1.「番号の分割入力」プロシージャ開始
2. Range型のオブジェクト変数「Myrange」を使用することを宣言
3. セル範囲【A3:A22】のすべてのセルに対して処理を繰り返す
4. オブジェクト変数「Myrange」が参照する5列右のセルに、セルの文字列から4文字分を左端から取り出し入力
5. オブジェクト変数「Myrange」が参照する6列右のセルに、セルの文字列から6文字目から1文字を取り出し入力
6. オブジェクト変数「Myrange」が参照する7列右のセルに、セルの文字列から2文字分を右端から取り出し入力
7. オブジェクト変数「Myrange」に次のセルへの参照を代入し、3行目に戻る
8. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。

※プロシージャの動作を確認します。

5-5 文字列を指定の種類に変換するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「プラン名の表記統一」プロシージャ

```
1. Sub プラン名の表記統一()
2.     Dim Myrange As Range
3.     For Each Myrange In Range("C3:C22")
4.         Myrange.Value = StrConv(Myrange.Value, vbProperCase)
5.     Next Myrange
6. End Sub
```

■プロシージャの意味

1.「プラン名の表記統一」プロシージャ開始
2. Range型のオブジェクト変数「Myrange」を使用することを宣言
3. セル範囲【C3:C22】のすべてのセルに対して処理を繰り返す
4. オブジェクト変数「Myrange」が参照するセルに、セルの文字列の先頭を大文字、それ以降を小文字に変換した文字列を入力
5. オブジェクト変数「Myrange」に次のセルへの参照を代入し、3行目に戻る
6. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

5-6 日付から年月日を取り出すには？

❶ 次のようにプロシージャを入力します。

※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「終了日の表示」プロシージャ

```
1. Sub 終了日の表示()
2.     Dim endYear As Integer
3.     Dim endMonth As Byte
4.     Dim endDay As Byte
5.     Dim mukouDate As Date
6.     endYear = Year(Range("E3").Value)
7.     endMonth = Month(Range("E3").Value)
8.     endDay = Day(Range("E3").Value)
9.     mukouDate = DateSerial(endYear, endMonth, endDay) + 1
10.    MsgBox "終了日は、" & endYear & "年" & endMonth & "月" & endDay & "日です。" &
       Chr(10) & _
11.    "(利用できなくなる年月日:" & mukouDate & ")"
12. End Sub
```

■プロシージャの意味

```
1.「終了日の表示」プロシージャ開始
2.     整数型の変数「endYear」を使用することを宣言
3.     バイト型の変数「endMonth」を使用することを宣言
4.     バイト型の変数「endDay」を使用することを宣言
5.     日付型の変数「mukouDate」を使用することを宣言
6.     変数「endYear」に、セル【E3】の日付から年を取り出して代入
7.     変数「endMonth」に、セル【E3】の日付から月を取り出して代入
8.     変数「endDay」に、セル【E3】の日付から日を取り出して代入
9.     変数「mukouDate」に、変数「endYear」「endMonth」「endDay」から取り出した日付の翌日の
       日付を代入
10.    変数「endYear」「endMonth」「endDay」と他の文字列、改行を連結し、
11.    変数「mukouDate」と他の文字列を連結してメッセージを表示
12. プロシージャ終了
```

※コンパイルを実行し、上書き保存しておきましょう。

※プロシージャの動作を確認します。

※10行目はコードが長いので、行継続文字「_（半角スペース+半角アンダースコア）」を使って行を複数に分割しています。行継続文字を使わずに1行で記述してもかまいません。

5-7 メッセージボックス内でタブや改行を入力するには？

❶ 次のようにプロシージャを入力します。

※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「申し込み情報の表示」プロシージャ

```
1. Sub 申し込み情報の表示()
2.     Dim torihikiBangou As String
3.     Dim className As String
4.     Dim planName As String
5.     Dim koushiName As String
6.     Dim endDate As Date
7.     torihikiBangou = Range("A3").Value
8.     className = Range("B3").Value
9.     planName = Range("C3").Value
10.    koushiName = Range("D3").Value
11.    endDate = Range("E3").Value
12.    MsgBox "取引番号" & Chr(9) & ":" & torihikiBangou & Chr(10) & _
13.        "クラス名" & Chr(9) & ":" & className & Chr(10) & _
14.        "プラン名" & Chr(9) & ":" & planName & Chr(10) & _
15         "講師名" & Chr(9) & ":" & koushiName & Chr(10) & _
16.        "終了日" & Chr(9) & ":" & endDate
17. End Sub
```

■プロシージャの意味

```
1.「申し込み情報の表示」プロシージャ開始
2.     文字列型の変数「torihikiBangou」を使用することを宣言
3.     文字列型の変数「className」を使用することを宣言
4.     文字列型の変数「planName」を使用することを宣言
5.     文字列型の変数「koushiName」を使用することを宣言
6.     日付型の変数「endDate」を使用することを宣言
7.     変数「torihikiBangou」にセル【A3】の値を代入
8.     変数「className」にセル【B3】の値を代入
9.     変数「planName」にセル【C3】の値を代入
10.    変数「koushiName」にセル【D3】の値を代入
11.    変数「endDate」にセル【E3】の値を代入
12.    変数「torihikiBangou」と他の文字列、タブと改行を連結し、
13.        変数「className」と他の文字列、タブと改行を連結し、
14.        変数「planName」と他の文字列、タブと改行を連結し、
15.        変数「koushiName」と他の文字列、タブと改行を連結し、
16.        変数「endDate」と他の文字列、タブを連結してメッセージを表示
17. プロシージャ終了
```

※コンパイルを実行し、上書き保存しておきましょう。

※プロシージャの動作を確認します。

※12～16行目はコードが長いので、行継続文字「_（半角スペース+半角アンダースコア）」を使って行を複数に分割しています。行継続文字を使わずに1行で記述してもかまいません。

5-8 入力可能なダイアログボックスを表示するには？

❶ 次のようにプロシージャを入力します。

※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「プラン名の入力」プロシージャ

```
1. Sub プラン名の入力()
2.     Dim planName As String
3.     planName = InputBox("プラン名を「Basic」か「Advanced」のいずれかで入力してください")
4.     If planName = "Basic" Or planName = "Advanced" Then
5.         Range("C22").Value = planName
6.     Else
7.         MsgBox "入力が間違っています。「Basic」か「Advanced」のいずれかを入力してください"
8.     End If
9. End Sub
```

■プロシージャの意味

```
1.「プラン名の入力」プロシージャ開始
2.     文字列型の変数「planName」を使用することを宣言
3.     変数「planName」に、メッセージ「プラン名を「Basic」か「Advanced」のいずれかで入力してください」と入力されたテキストボックスを表示し、入力された値を代入
4.     変数「planName」の値が「Basic」か「Advanced」の場合は
5.         セル【C22】に変数「planName」の値を代入
6.     それ以外の場合は
7.         「入力が間違っています。「Basic」か「Advanced」のいずれかを入力してください」とメッセージを表示
8.     Ifステートメント終了
9. プロシージャ終了
```

※コンパイルを実行し、上書き保存しておきましょう。

※プロシージャの動作を確認します。

5-9 指定した表示形式を設定するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「料金の表示形式の設定」プロシージャ

```
1. Sub 料金の表示形式の設定()
2.     Dim Myrange As Range
3.     For Each Myrange In Range("K4:K5")
4.         Myrange.Value = Format(Myrange.Value, "#,###円")
5.     Next Myrange
6. End Sub
```

■プロシージャの意味

```
1.「料金の表示形式の設定」プロシージャ開始
2.     Range型のオブジェクト変数「Myrange」を使用することを宣言
3.     セル範囲【K4:K5】のすべてのセルに対して処理を繰り返す
4.         オブジェクト変数「Myrange」が参照するセルに、表示形式を「#,###円」に設定して入力
5.     オブジェクト変数「Myrange」に次のセルへの参照を代入し、3行目に戻る
6. プロシージャ終了
```

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

5-10 指定した値の種類を判断するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「予約番号の背景色を設定」プロシージャ

```
1. Sub 予約番号の背景色を設定()
2.     Dim Myrange As Range
3.     For Each Myrange In Range("A3:A12")
4.         If Not IsNumeric(Myrange.Value) Then
5.             Myrange.Interior.Color = vbGreen
6.         End If
7.     Next Myrange
8. End Sub
```

■プロシージャの意味

```
1.「予約番号の背景色を設定」プロシージャ開始
2.     Range型のオブジェクト変数「Myrange」を使用することを宣言
3.     セル範囲【A3:A12】のすべてのセルに対して処理を繰り返す
4.         オブジェクト変数「Myrange」が参照するセルの値が数値でない場合は
5.             オブジェクト変数「Myrange」が参照するセルの背景色を緑色に設定
6.         Ifステートメント終了
7.     オブジェクト変数「Myrange」に次のセルへの参照を代入し、3行目に戻る
8. プロシージャ終了
```

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

5-11 文字列を区切ったり結合したりするには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「名前の個別入力」プロシージャ

```
1. Sub 名前の個別入力()
2.     Dim Item As Variant
3.     Dim i As Byte
4.     i = 4
5.     For Each Item In Split(Range("B3").Value, ",")
6.         Range("H" & i).Value = Item
7.         i = i + 1
8.     Next Item
9. End Sub
```

■プロシージャの意味

1.「名前の個別入力」プロシージャ開始
2.　　バリアント型の変数「Item」を使用することを宣言
3.　　バイト型の変数「i」を使用することを宣言
4.　　変数「i」に「4」を代入
5.　　区切り文字「,」で分割したセル【B3】の文字列に対して処理を繰り返す
6.　　H列の変数「i」行のセルに、変数「Item」の値を設定
7.　　変数「i」に、変数「i」＋1の結果を代入
8.　　変数「Item」に次の要素への参照を代入し、5行目に戻る
9. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

5-12 条件式を満たすかどうかで異なる値を返すには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「団体割適用の入力」プロシージャ

```
1. Sub 団体割適用の入力()
2.     Dim Myrange As Range
3.     For Each Myrange In Range("E3:E12")
4.         Myrange.Value = IIf(Myrange.Offset(0, -2).Value + Myrange.Offset(0, -1).
           Value >= 5, "団体割", "")
5.     Next Myrange
6. End Sub
```

■プロシージャの意味

1.「団体割適用の入力」プロシージャ開始
2.　Range型のオブジェクト変数「Myrange」を使用することを宣言
3.　セル範囲【E3:E12】のすべてのセルに対して処理を繰り返す
4.　　オブジェクト変数「Myrange」が参照するセルに、2列左に移動したセルの値と、1列左に移動したセルの値を合計した値が5以上の場合は「団体割」、それ以外の場合は空の文字列を入力
5.　オブジェクト変数「Myrange」に次のセルへの参照を代入し、3行目に戻る
6. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

5-13 ワークシート関数をプロシージャ内で利用するには？

❶ 次のようにプロシージャを入力します。

※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「予約者数の計算」プロシージャ

```
1. Sub 予約者数の計算()
2.     Dim total As Integer
3.     Dim totalAdults As Integer
4.     Dim totalChildren As Integer
5.     total = WorksheetFunction.Sum(Range("C3:D12"))
6.     totalAdults = WorksheetFunction.Sum(Range("C3:C12"))
7.     totalChildren = WorksheetFunction.Sum(Range("D3:D12"))
8.     Range("D14").Value = total
9.     Range("D15").Value = Format(totalAdults / total, "0.00%")
10.    Range("D16").Value = Format(totalChildren / total, "0.00%")
11. End Sub
```

■プロシージャの意味

1.「予約者数の計算」プロシージャ開始
2. 整数型の変数「total」を使用することを宣言
3. 整数型の変数「totalAdults」を使用することを宣言
4. 整数型の変数「totalChildren」を使用することを宣言
5. 変数「total」に、SUM関数でセル範囲【C3:D12】の合計値を設定
6. 変数「totalAdults」に、SUM関数でセル範囲【C3:C12】の合計値を設定
7. 変数「totalChildren」に、SUM関数でセル範囲【D3:D12】の合計値を設定
8. セル【D14】に、変数「total」の値を入力
9. セル【D15】に、変数「totalAdults」を変数「total」で割った値を、表示形式を「0.00%」に設定して入力
10. セル【D16】に、変数「totalChildren」を変数「total」で割った値を、表示形式を「0.00%」に設定して入力
11. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。

※プロシージャの動作を確認します。

5-14 ユーザー定義関数を作成・利用するには？

❶ 次のようにプロシージャを入力します。

※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「団体割の計算」プロシージャ

```
1. Function 団体割の計算(大人の人数, 子供の人数)
2.     団体割の計算 = (大人の人数 * 2500 + 子供の人数 * 1500) * 0.8
3. End Function
```

■プロシージャの意味

1.「団体割の計算」プロシージャ開始(引数に「大人の人数」と「子供の人数」を指定)
2.　「団体割の計算」に「(大人の人数×2500+子供の人数×1500)×0.8」の計算結果を代入
3. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。

❷ セル【J9】をクリックします。

❸ fx(関数の挿入)をクリックします。

《関数の挿入》ダイアログボックスが表示されます。

❹《関数の分類》の⌄をクリックし、一覧から《ユーザー定義》を選択します。

❺《関数名》の一覧から、「団体割の計算」を選択します。

❻《OK》をクリックします。

《関数の引数》ダイアログボックスが表示されます。

❼《大人の人数》のテキストボックスにカーソルがあることを確認します。

❽ セル【C3】をクリックします。

❾《子供の人数》のテキストボックスをクリックして、カーソルを表示します。

❿ セル【D3】をクリックします。

⓫《OK》をクリックします。

ユーザー定義関数が実行されます。

※数式バーに指定したユーザー定義関数が入力されます。

第6章 イベントの利用

6-1 シートがアクティブになったときに処理を行うには？

※VBEを起動しておきましょう。

❶ プロジェクトエクスプローラーの《Sheet1（管理表）》をダブルクリックします。

コードウィンドウに《Sheet1》オブジェクトモジュールの内容が表示されます。

❷ 《オブジェクト》ボックスの  をクリックし、一覧から《Worksheet》を選択します。

「Worksheet_SelectionChange」イベントプロシージャが作成されます。

❸ 《プロシージャ》ボックスの  をクリックし、一覧から《Activate》を選択します。

「Worksheet_Activate」イベントプロシージャが作成されます。

※「Worksheet_SelectionChange」イベントプロシージャは削除しておきましょう。

❹ 次のように「Worksheet_Activate」イベントプロシージャを入力します。

■「Worksheet_Activate」イベントプロシージャ

```
1. Private Sub Worksheet_Activate()
2.     MsgBox "管理者以外は編集しないでください"
3. End Sub
```

■プロシージャの意味

1.「Worksheet_Activate」イベントプロシージャ開始
2.　　「管理者以外は編集しないでください」とメッセージを表示
3. プロシージャ終了

※コンパイルを実行し上書き保存して、Excelに切り替えておきましょう。

❺ ワークシート「管理表」を選択します。

※メッセージボックスが表示されることを確認します。

6-2 選択範囲を変更したときに処理を行うには？

※VBEを起動しておきましょう。

❶ プロジェクトエクスプローラーの《Sheet2（7月）》をダブルクリックします。

コードウィンドウに《Sheet2》オブジェクトモジュールの内容が表示されます。

❷ 《オブジェクト》ボックスの  をクリックし、一覧から《Worksheet》を選択します。

「Worksheet_SelectionChange」イベントプロシージャが作成されます。

❸ 次のように「Worksheet_SelectionChange」イベントプロシージャを入力します。

■「Worksheet_SelectionChange」イベントプロシージャ

```
1. Private Sub Worksheet_SelectionChange(ByVal Target As Range)
2.     Dim Myrange As Range
3.     Set Myrange = Application.Intersect(Target, Range("D6:E36"))
4.     If Not Myrange Is Nothing Then
5.         Myrange.Value = Time
6.     End If
7.     Set Myrange = Nothing
8. End Sub
```

■プロシージャの意味

1.「Worksheet_SelectionChange（Range型の引数Targetは選択したセル範囲）」イベントプロシージャ開始
2. Range型のオブジェクト変数「Myrange」を使用することを宣言
3. オブジェクト変数「Myrange」に、引数Targetが参照するセル範囲とセル範囲【D6:E36】の共有セルへの参照を代入
4. オブジェクト変数「Myrange」がNothingでない場合（共有セルが選択された場合）
5. 現在の時刻をオブジェクト変数「Myrange」が参照するセルに入力
6. Ifステートメント終了
7. オブジェクト変数「Myrange」を初期化
8. イベントプロシージャ終了

※コンパイルを実行し上書き保存して、Excelに切り替えておきましょう。

❹ セル範囲【D6:E36】の任意のセルを選択します。

※現在の時刻が入力されることを確認します。

6-3 セルをダブルクリックしたときに処理を行うには？

※VBEを起動しておきましょう。

❶ プロジェクトエクスプローラーの《Sheet2（7月）》をダブルクリックします。

コードウィンドウに《Sheet2》オブジェクトモジュールの内容が表示されます。

❷ 《オブジェクト》ボックスの  をクリックし、一覧から《Worksheet》を選択します。

「Worksheet_SelectionChange」イベントプロシージャが作成されます。

❸ 《プロシージャ》ボックスの  をクリックし、一覧から《BeforeDoubleClick》を選択します。

「Worksheet_BeforeDoubleClick」イベントプロシージャが作成されます。

※「Worksheet_SelectionChange」イベントプロシージャは削除しておきましょう。

❹ 次のように「Worksheet_BeforeDoubleClick」イベントプロシージャを入力します。

■「Worksheet_BeforeDoubleClick」イベントプロシージャ

```
1. Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Range, Cancel As Boolean)
2.     Dim Myrange As Range
3.     Set Myrange = Application.Intersect(Target, Range("G6:G36"))
4.     If Not Myrange Is Nothing Then
5.         Myrange.Value = "休み"
6.         Cancel = True
7.     End If
8.     Set Myrange = Nothing
9. End Sub
```

■プロシージャの意味

```
1.「Worksheet_BeforeDoubleClick(Range型の引数Targetはダブルクリックしたセル、ブール型の
   引数Cancel)」イベントプロシージャ開始
2.     Range型のオブジェクト変数「Myrange」を使用することを宣言
3.     オブジェクト変数「Myrange」に、引数Targetが参照するセルとセル範囲【G6:G36】の共有セル
       への参照を代入
4.     オブジェクト変数「Myrange」がNothingでない場合（共有セルが選択された場合）
5.         オブジェクト変数「Myrange」が参照するセルに「休み」と入力
6.         引数CancelにTrueを代入（編集モードをキャンセル）
7.     Ifステートメント終了
8.     オブジェクト変数「Myrange」を初期化
9. イベントプロシージャ終了
```

※コンパイルを実行し上書き保存し、Excelに切り替えておきましょう。

❺ セル範囲【G6：G36】の任意のセルを選択します。

※「休み」と入力されることを確認します。

6-4 ブックを開くとき・閉じる前に処理を行うには?

※VBEを起動しておきましょう。

まずは、「Workbook_Open」イベントプロシージャを作成します。

❶ プロジェクトエクスプローラーの《ThisWorkbook》をダブルクリックします。

コードウィンドウに《ThisWorkbook》オブジェクトモジュールの内容が表示されます。

❷ 《オブジェクト》ボックスの  をクリックし、一覧から《Workbook》を選択します。

「Workbook_Open」イベントプロシージャが作成されます。

❸ 次のように「Workbook_Open」イベントプロシージャを入力します。

■「Workbook_Open」イベントプロシージャ

```
1. Private Sub Workbook_Open()
2.     MsgBox "出勤時刻・退勤時刻を打刻しましょう"
3. End Sub
```

■プロシージャの意味

1.「Workbook_Open」イベントプロシージャ開始
2.　　「出勤時刻・退勤時刻を打刻しましょう」とメッセージを表示
3. イベントプロシージャ終了

続いて、「Workbook_BeforeClose」イベントプロシージャを作成します。

❹ 《プロシージャ》ボックスの  をクリックし、一覧から《BeforeClose》を選択します。

「Workbook_BeforeClose」イベントプロシージャが作成されます。

❺ 次のように「Workbook_BeforeClose」イベントプロシージャを入力します。

■「Workbook_BeforeClose」イベントプロシージャ

```
1. Private Sub Workbook_BeforeClose(Cancel As Boolean)
2.     If MsgBox("打刻漏れがないか確認しましたか？", vbYesNo) = vbYes Then
3.         ThisWorkbook.Save
4.     Else
5.         Cancel = True
6.     End If
7. End Sub
```

■プロシージャの意味

1.「Workbook_BeforeClose(ブール型の引数Cancel)」イベントプロシージャ開始
2.　　「はい」「いいえ」ボタンを持つメッセージボックスに「打刻漏れがないか確認しましたか？」と表示し、「はい」がクリックされた場合は
3.　　実行中のプロシージャが記述されているブックを上書き保存
4.　　それ以外の場合は
5.　　引数CancelにTrueを代入(ブックを閉じる操作をキャンセル)
6.　　Ifステートメント終了
7. イベントプロシージャ終了

※コンパイルを実行し上書き保存して、Excelに切り替えておきましょう。

❻ ブックを閉じます。

※「打刻漏れがないか確認しましたか?」とメッセージが表示されることを確認します。

❼ 再度ブックを開きます。

※「出勤時刻・退勤時刻を打刻しましょう」とメッセージが表示されることを確認します。

6-5 シートを作成したときに処理を行うには？

※VBEを起動しておきましょう。

❶ プロジェクトエクスプローラーの《ThisWorkbook》をダブルクリックします。

コードウィンドウに《ThisWorkbook》オブジェクトモジュールの内容が表示されます。

❷ 《オブジェクト》ボックスの をクリックし、一覧から《Workbook》を選択します。

「Workbook_Open」イベントプロシージャが作成されます。

❸ 《プロシージャ》ボックスの をクリックし、一覧から《NewSheet》を選択します。

「Workbook_NewSheet」イベントプロシージャが作成されます。

※「Workbook_Open」イベントプロシージャは削除しておきましょう。

❹ 次のように「Workbook_NewSheet」イベントプロシージャを入力します。

■「Workbook_NewSheet」イベントプロシージャ

```
1. Private Sub Workbook_NewSheet(ByVal Sh As Object)
2.     Dim newSheetName As String
3.     newSheetName = InputBox("シートの名前を入力してください。")
4.     If newSheetName <> "" Then
5.         Sh.Name = newSheetName
6.     Else
7.         MsgBox "シート名が空です。デフォルト名を使用します。"
8.     End If
9. End Sub
```

■プロシージャの意味

```
1.「Workbook_NewSheet(Object型の引数Shは作成されたシート)」イベントプロシージャ開始
2.     文字列型の変数「newSheetName」を使用することを宣言
3.     変数「newSheetName」に、InputBoxに入力された値を代入
4.     変数「newSheetName」の値が空文字(「""」)でない場合(文字列が入力された場合)は
5.         新しいシートの名前に変数「newSheetName」の値を設定
6.     それ以外の場合は
7.         「シート名が空です。デフォルト名を使用します。」とメッセージを表示
8.     Ifステートメント終了
9. イベントプロシージャ終了
```

※コンパイルを実行し上書き保存して、Excelに切り替えておきましょう。

❺ 新しいシートを作成します。

※シート名を入力するダイアログボックスが表示されることを確認します。

❻ 「8月」と入力します。

❼ 《OK》をクリックします。

※新しく追加したワークシートの名前が「8月」に変わることを確認します。

※シート名を空白にして《OK》をクリックした場合は、メッセージが表示されることも確認しておきましょう。

第7章 エラー処理・デバッグ

7-1 コンパイルエラーを修正するには？

※VBEを起動し、《標準モジュール》→「Module1」を開いておきましょう。

❶ 次のように「**合計金額**」プロシージャに入力します。

■「合計金額」プロシージャ

```
1. Sub 合計金額()
2.     Dim kingaku As Long
3.     kingaku = Worksheet("講座開催状況").Range("K3").Value
4. End Sub
```

❷ 《デバッグ》をクリックします。

❸ 《VBAProjectのコンパイル》をクリックします。

コンパイルエラーのエラーメッセージが表示されます。

❹ 《OK》をクリックします。

❺ 3行目のステートメントを次のように修正します。

```
kingaku = Worksheets("講座開催状況").Range("K3").Value
```

※再度コンパイルを実行してエラーが発生しないことを確認します。確認後、上書き保存しておきましょう。

7-2 実行時エラーを修正するには？

※VBEを起動し、《標準モジュール》→「Module1」を開いておきましょう。

❶「並べ替え」プロシージャにカーソルがある状態で、  （Sub/ユーザーフォームの実行）をクリックします。

ダイアログボックスが表示されます。任意の値を入力すると、「型が一致しません」と実行時エラーが発生し、処理が中断します。

❷《デバッグ》をクリックします。

インデントマーカーの場所をチェックし、整数型の変数に文字列を代入しようとしていることを確認します。

❸ 3行目を次のように修正します。

```
Dim sortcell As String
```

❹  （リセット）をクリックします。

※再度プロシージャを実行してエラーが発生しないことを確認します。確認後、上書き保存しておきましょう。

7-3 実行時エラーが発生しても処理を継続するには？

※VBEを起動し、《標準モジュール》→「Module1」を開いておきましょう。

❶「原価率計算」プロシージャにカーソルがある状態で、  （Sub/ユーザーフォームの実行）をクリックします。

「0で除算しました」と実行時エラーが発生し、処理が中断します。

❷《終了》をクリックします。

❸「原価率計算」プロシージャを、次のように修正します。

■「原価率計算」プロシージャ

```
1. Sub 原価率計算()
2.     Dim kakaku As Double
3.     Dim genka As Double
4.     Dim ritu As Double
5.     Dim i As Integer
6.     On Error Resume Next
7.     For i = 0 To 9
8.         kakaku = 0
9.         genka = 0
10.        kakaku = Range("B4").Offset(i, 0).Value
11.        genka = Range("C4").Offset(i, 0).Value
12.        ritu = genka / kakaku * 100
13.        Range("D4").Offset(i, 0).Value = ritu
14.    Next
15. End Sub
```

■プロシージャの意味

```
1.「原価率計算」プロシージャ開始
2.     倍精度浮動小数点数型の変数「kakaku」を使用することを宣言
3.     倍精度浮動小数点数型の変数「genka」を使用することを宣言
4.     倍精度浮動小数点数型の変数「ritu」を使用することを宣言
5.     整数型の変数「i」を使用することを宣言
6.     エラー処理を開始（実行時エラーが発生しても処理を継続）
7.     変数「i」が「0」から「9」まで繰り返す処理を開始
8.         変数「kakaku」に「0」を代入
9.         変数「genka」に「0」を代入
10.        セル【B4】から変数「i」だけ下に移動したセルの値を変数「kakaku」に代入
11.        セル【C4】から変数「i」だけ下に移動したセルの値を変数「genka」に代入
12.        変数「genka」/ 変数「kakaku」* 100の結果を変数「ritu」に代入
13.        セル【D4】から変数「i」だけ下に移動したセルに変数「ritu」の値を代入
14.    変数「i」に変数「i」+1の結果を代入し、7行目に戻る
15.プロシージャ終了
```

※再度「原価率計算」プロシージャを実行します。エラーが発生せず、最後まで実行されることを確認します。確認後、上書き保存しておきましょう。

7-4 実行時エラーの発生時に別の処理を実行するには？

※VBEを起動し、《標準モジュール》→「Module1」を開いておきましょう。

❶「原価率計算」プロシージャを、次のように修正します。

■「原価率計算」プロシージャ

```
1. Sub 原価率計算()
2.     Dim kakaku As Double
3.     Dim genka As Double
4.     Dim ritu As Double
5.     Dim i As Integer
6.     On Error GoTo ErrorKakaku
7.     For i = 0 To 9
8.         kakaku = 0
9.         genka = 0
10.        kakaku = Range("B4").Offset(i, 0).Value
11.        genka = Range("C4").Offset(i, 0).Value
12.        ritu = genka / kakaku * 100
13.        Range("D4").Offset(i, 0).Value = ritu
14.    Next
15.    Exit Sub
16. ErrorKakaku:
17.    MsgBox "0による除算はできません。" & Chr(10) & "処理を継続します。"
18.    kakaku = genka * 2
19.    Resume
20. End Sub
```

■プロシージャの意味

```
1.「原価率計算」プロシージャ開始
2.     倍精度浮動小数点数型の変数「kakaku」を使用することを宣言
3.     倍精度浮動小数点数型の変数「genka」を使用することを宣言
4.     倍精度浮動小数点数型の変数「ritu」を使用することを宣言
5.     整数型の変数「i」を使用することを宣言
6.     エラー処理を開始（実行時エラーが発生したら「ErrorKakaku」に制御を移す）
7.     変数「i」が「0」から「9」になるまで次の行以降の処理を繰り返す
8.         変数「kakaku」に「0」を代入
9.         変数「genka」に「0」を代入
10.        セル【B4】から変数「i」だけ下に移動したセルの値を変数「kakaku」に代入
11.        セル【C4】から変数「i」だけ下に移動したセルの値を変数「genka」に代入
12.        変数「genka」/変数「kakaku」＊100の結果を変数「ritu」に代入
13.        セル【D4】から変数「i」だけ下に移動したセルに変数「ritu」の値を代入
14.    変数「i」に変数「i」＋1の結果を代入し、7行目に戻る
15.    プロシージャを抜け出す
16. エラー処理ルーチン（行ラベル ErrorKakaku）
17.    メッセージを表示
18.    変数「kakaku」に変数「genka」を2倍した値を代入
19.    7行目に制御を戻す
20. プロシージャ終了
```

※再度「原価率計算」プロシージャを実行します。メッセージが2回表示され、エラーが発生せず、最後まで実行されることを確認します。確認後、上書き保存しておきましょう。

第8章 ユーザーフォームの利用

8-1 ユーザーフォームを追加・表示するには？

※VBEを起動しておきましょう。

❶《挿入》をクリックします。

❷《ユーザーフォーム》をクリックします。

新しいユーザーフォームが作成されます。

❸ プロジェクトエクスプローラーの「UserForm1」をクリックします。

プロパティウィンドウの《オブジェクト》ボックスに「UserForm1 UserForm」と表示されます。

❹ プロパティウィンドウの《(オブジェクト名)》をクリックします。

❺《(オブジェクト名)》の設定値に「受注入力」と入力し、Enterを押します。

プロパティウィンドウの《オブジェクト》ボックスが「受注入力　UserForm」に変わります。

❻ プロパティウィンドウの《Caption》をクリックします。

❼《Caption》の設定値に「受注情報の入力」と入力し、Enterを押します。

ユーザーフォームのタイトルバーに「受注情報の入力」と表示されます。

❽ ユーザーフォームのタイトルバーをクリックします。

❾ (Sub/ユーザーフォームの実行) をクリックします。

Excelに切り替わり、ユーザーフォームが表示されます。

※ユーザーフォームを閉じておきましょう。ユーザーフォームを閉じると、VBEに切り替わります。

※上書き保存しておきましょう。

8-2 コマンドボタンを追加するには？

※VBEを起動しておきましょう。

❶ プロジェクトエクスプローラーのフォーム「**受注入力**」をダブルクリックします。

❷《**ツールボックス**》の  （コマンドボタン）をクリックします。

❸ 左側のコマンドボタンを追加する場所をポイントします。

マウスポインターの形が  に変わります。

❹ クリックします。

既定のサイズのコマンドボタンが追加されます。

❺ コマンドボタンが選択されていることを確認します。

❻ Ctrl を押しながらコマンドボタンを右側にドラッグします。

コマンドボタンがコピーされます。

❼ 左側の「CommandButton1」コマンドボタンを選択します。

❽ プロパティウィンドウの《(**オブジェクト名**)》をクリックします。

❾《(**オブジェクト名**)》の設定値に「cmdSend」と入力し、Enter を押します。

プロパティウィンドウの《**オブジェクト**》ボックスが「cmdSend CommandButton」に変わります。

❿ プロパティウィンドウの《Caption》をクリックします。

⓫《Caption》の設定値に「**送信**」と入力し、Enter を押します。

ユーザーフォームの左側のコマンドボタンに「**送信**」と表示されます。

⓬ 同様に、右側の「CommandButton2」コマンドボタンのプロパティを設定します。

※ユーザーフォームを実行して結果を確認しておきましょう。VBEに切り替え、上書き保存しておきましょう。

8-3 ラベルを追加するには？

※VBEを起動しておきましょう。

❶ プロジェクトエクスプローラーのフォーム「受注入力」をダブルクリックします。

❷《ツールボックス》の  (ラベル) をクリックします。

❸ 上から1番目のラベルを追加する場所をポイントします。

マウスポインターの形が  に変わります。

❹ 右下方向にドラッグします。

任意のサイズのラベルが追加されます。

❺ プロパティウィンドウの《Caption》をクリックします。

❻《Caption》の設定値に「商品コード」と入力し、Enter を押します。

ユーザーフォームのラベルに「商品コード」と表示されます。

❼ 同様に残りのラベルを追加して、プロパティを設定します。

※ラベルの追加はコピーを使うと効率的です。

※ユーザーフォームを実行して結果を確認しておきましょう。VBEに切り替え、上書き保存しておきましょう。

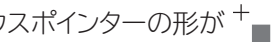
8-4 テキストボックスを追加するには？

※VBEを起動しておきましょう。

❶ プロジェクトエクスプローラーのフォーム「受注入力」をダブルクリックします。

❷《ツールボックス》の  （テキストボックス）をクリックします。

❸ テキストボックスを追加する場所をポイントします。

マウスポインターの形が  に変わります。

❹ クリックします。

既定のサイズのテキストボックスが追加されます。

❺ プロパティウィンドウの《(オブジェクト名)》をクリックします。

❻《(オブジェクト名)》の設定値に「txtShohinCode」と入力し、Enterを押します。

プロパティウィンドウの《オブジェクト》ボックスが「txtShohinCode TextBox」に変わります。

❼ プロパティウィンドウの《IMEMode》をクリックします。

❽《IMEMode》の設定値の  をクリックし、「2-fmIMEModeOff」を選択します。

※ユーザーフォームを実行して結果を確認しておきましょう。VBEに切り替え、上書き保存しておきましょう。

8-5 オプションボタンを追加するには？

※VBEを起動しておきましょう。

❶ プロジェクトエクスプローラーのフォーム「受注入力」をダブルクリックします。

❷《ツールボックス》の  （オプションボタン）をクリックします。

❸ 左側のオプションボタンを追加する場所をポイントします。

マウスポインターの形が  に変わります。

❹ 右下方向にドラッグします。

任意のサイズのオプションボタンが追加されます。

❺「OptionButton1」オプションボタンが選択されていることを確認します。

❻ Ctrl を押しながらオプションボタンを右側にドラッグします。

オプションボタンがコピーされます。

❼「OptionButton1」オプションボタンを選択します。

❽ プロパティウィンドウの《(オブジェクト名)》をクリックします。

❾《(オブジェクト名)》の設定値に「opt1」と入力し、Enter を押します。

プロパティウィンドウの《オブジェクト》ボックスが「opt1 OptionButton」に変わります。

❿ プロパティウィンドウの《Caption》をクリックします。

⓫《Caption》の設定値に「有」と入力し、Enter を押します。

ユーザーフォームのオプションボタンに「有」と表示されます。

⓬ プロパティウィンドウの《Value》の設定値が「False」になっていることを確認します。

⓭ 同様に、「OptionButton2」オプションボタンのプロパティを設定します。

※ユーザーフォームを実行して結果を確認しておきましょう。VBEに切り替え、上書き保存しておきましょう。

8-6 コンボボックスを追加するには？

※VBEを起動しておきましょう。

❶ プロジェクトエクスプローラーのフォーム「受注入力」をダブルクリックします。

❷《ツールボックス》の  （リストボックス）をクリックします。

❸ リストボックスを追加する場所をポイントします。

マウスポインターの形が ＋■ に変わります。

❹ クリックします。

既定のサイズのリストボックスが追加されます。

❺ プロパティウィンドウの《（オブジェクト名）》をクリックします。

❻《（オブジェクト名）》の設定値に「lstShiharai」と入力し、Enter を押します。

プロパティウィンドウの《オブジェクト》ボックスが「lstShiharai ListBox」に変わります。

❼ プロパティウィンドウの《RowSource》をクリックします。

❽《RowSource》の設定値に「支払方法」と入力し、Enter を押します。

支払方法のリストが表示されます。

※すべての項目が表示されるように、リストボックスのサイズを調整しておきましょう。

※ユーザーフォームを実行して結果を確認しておきましょう。VBEに切り替え、上書き保存しておきましょう。

8-7 チェックボックスを追加するには？

※VBEを起動しておきましょう。

❶ プロジェクトエクスプローラーのフォーム「受注入力」をダブルクリックします。

❷《ツールボックス》の  （チェックボックス）をクリックします。

❸ チェックボックスを追加する場所でドラッグします。

任意のサイズのチェックボックスが追加されます。

❹ プロパティウィンドウの《（オブジェクト名）》をクリックします。

❺《（オブジェクト名）》の設定値に「chk1」と入力し、Enter を押します。

❻ プロパティウィンドウの《Caption》をクリックします。

❼《Caption》の設定値に「発送完了時」と入力し、Enter を押します。

❽ 同様に、「CheckBox2」チェックボックスを右側に追加し、プロパティを設定します。

※ユーザーフォームを実行して結果を確認しておきましょう。VBEに切り替え、上書き保存しておきましょう。

8-8 ユーザーフォームの入力値をセルに反映するには?

※VBEを起動しておきましょう。

❶ ユーザーフォームの「cmdSend」コマンドボタンをダブルクリックします。

コードウィンドウが表示され、「cmdSend_Click」イベントプロシージャが作成されます。

❷ 次のように「cmdSend_Click」イベントプロシージャを入力します。

■「cmdSend_Click」イベントプロシージャ

```
Private Sub cmdSend_Click()
    Dim tsuchi As String
    Range("A3").Select
    Selection.End(xlDown).Select
    ActiveCell.Offset(1, 0).Value = txtShohinCode.Text
    ActiveCell.Offset(1, 1).Value = IIf(opt1.Value = True, "有", "無")
    ActiveCell.Offset(1, 2).Value = lstShiharai.Text
    Select Case True
        Case chk1.Value = True And chk2.Value = True
            tsuchi = "発送完了時&配達完了時"
        Case chk1.Value = True
            tsuchi = "発送完了時"
        Case chk2.Value = True
            tsuchi = "配達完了時"
        Case Else
            tsuchi = "なし"
        End Select
    ActiveCell.Offset(1, 3).Value = tsuchi
End Sub
```

■プロシージャの意味

```
「cmdSend_Click」イベントプロシージャ開始
    文字列型の変数「tsuchi」を使用することを宣言
    セル【A3】を選択
    Ctrl + ↓ でデータの下端のセルを選択
    アクティブセルの1行下のセルにtxtShohinCodeの値を入力
    アクティブセルの1行下、1列右のセルにopt1がオンの場合は「有」、オフの場合は「無」を入力
    アクティブセルの1行下、2列右のセルにlstShiharaiの値を入力
    チェックボックスの状態が
        chk1とchk2が両方ともオンの場合は
            変数「tsuchi」に「発送完了時&配達完了時」を代入
        chk1がオンの場合は
            変数「tsuchi」に「発送完了時」を代入
        chk2がオンの場合は
            変数「tsuchi」に「配達完了時」を代入
        それ以外の場合は
            変数「tsuchi」に「なし」を代入
        Select Caseステートメント終了
    アクティブセルの1行下、3列右のセルに変数「tsuchi」の値を入力
イベントプロシージャ終了
```

※コンパイルを実行し、上書き保存しておきましょう。ユーザーフォームを実行して、プロシージャの動作を確認します。

❸ ユーザーフォームのコントロールに、任意の値を設定し、「cmdSend」ボタンをクリックします。

※セル範囲【A11:D11】に、ユーザーフォームで入力した値が反映されることを確認します。

第9章 ファイルシステムオブジェクトの利用

9-1 フォルダーを操作するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「フォルダー作成削除」プロシージャ

```
1. Sub フォルダー作成削除()
2.     Dim MyFSO As New FileSystemObject
3.     Dim folderpath As String
4.     Dim folderpath2 As String
5.     folderpath = ThisWorkbook.Path & "¥9-1Practice"
6.     folderpath2 = ThisWorkbook.Path & "¥9-1Practice2"
7.     If MyFSO.FolderExists(folderpath) And MyFSO.FolderExists(folderpath2) Then
8.         MyFSO.DeleteFolder FolderSpec:=folderpath
9.         MyFSO.DeleteFolder FolderSpec:=folderpath2
10.    ElseIf MyFSO.FolderExists(folderpath) Then
11.        MyFSO.DeleteFolder FolderSpec:=folderpath
12.    ElseIf MyFSO.FolderExists(folderpath2) Then
13.        MyFSO.DeleteFolder FolderSpec:=folderpath2
14.    Else
15.        MyFSO.CreateFolder Path:=folderpath
16.        MyFSO.CreateFolder Path:=folderpath2
17.    End If
18.    Set MyFSO = Nothing
19. End Sub
```

■プロシージャの意味

```
1.「フォルダー作成削除」プロシージャ開始
2.     FileSystemObject型のオブジェクト変数「MyFSO」を使用することを宣言してインスタンスを
       生成
3.     文字列型の変数「folderpath」を使用することを宣言
4.     文字列型の変数「folderpath2」を使用することを宣言
5.     変数「folderpath」に実行中のプロシージャが記述されたブックが保存されているフォルダーの
       絶対パスと「¥9-1Practice」を連結して代入
6.     変数「folderpath2」に実行中のプロシージャが記述されたブックが保存されているフォルダーの
       絶対パスと「¥9-1Practice2」を連結して代入
7.     変数「folderpath」のフォルダーと変数「folderpath2」のフォルダーが存在する場合は
8.         変数「folderpath」のフォルダーを削除
9.         変数「folderpath2」のフォルダーを削除
10.    変数「folderpath」のフォルダーが存在する場合は
11.        変数「folderpath」のフォルダーを削除
12.    変数「folderpath2」のフォルダーが存在する場合は
13.        変数「folderpath2」のフォルダーを削除
14.    それ以外の場合は
15.        変数「folderpath」のフォルダーを作成
16.        変数「folderpath2」のフォルダーを作成
17.    Ifステートメント終了
18.    オブジェクト変数「MyFSO」の初期化
19. プロシージャ終了
```

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

9-2 ファイルを操作するには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「ファイルコピー削除」プロシージャ

```
1. Sub ファイルコピー削除()
2.     Dim MyFSO As New FileSystemObject
3.     Dim filename As String
4.     Dim filename2 As String
5.     filename = ThisWorkbook.Path & "¥9-2Practice.txt"
6.     filename2 = ThisWorkbook.Path & "¥9-2Practice2.txt"
7.     If MyFSO.FileExists(filename2) Then
8.         MyFSO.DeleteFile FileSpec:=filename2
9.         MsgBox "「" & filename2 & "」を削除しました"
10.    Else
11.        MyFSO.CopyFile Source:=filename, Destination:=filename2
12.        MsgBox "「" & filename2 & "」を作成しました"
13.    End If
14.    Set MyFSO = Nothing
15. End Sub
```

■プロシージャの意味

1.「ファイルコピー削除」プロシージャ開始
2. FileSystemObject型のオブジェクト変数「MyFSO」を使用することを宣言してインスタンスを生成
3. 文字列型の変数「filename」を使用することを宣言
4. 文字列型の変数「filename2」を使用することを宣言
5. 変数「filename」に実行中のプロシージャが記述されたブックが保存されているフォルダーの絶対パスと「¥9-2Practice.txt」を連結して代入
6. 変数「filename2」に実行中のプロシージャが記述されたブックが保存されているフォルダーの絶対パスと「¥9-2Practice2.txt」を連結して代入
7. 変数「filename2」のファイルが存在する場合は
8. 変数「filename2」のファイルを削除
9. 変数「filename2」と他の文字列を連結してメッセージを表示
10. それ以外の場合は
11. 変数「filename」のファイルを変数「filename2」の場所とファイル名でコピー
12. 変数「filename2」と他の文字列を連結してメッセージを表示
13. Ifステートメント終了
14. オブジェクト変数「MyFSO」の初期化
15. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

9-3 テキストファイルを読み込むには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「ラベルテキスト読込」プロシージャ

```
1. Sub ラベルテキスト読込()
2.     Dim MyFSO As New FileSystemObject
3.     Dim MyTXT As TextStream
4.     Dim filename As String
5.     filename = ThisWorkbook.Path & "¥9-3Practice.txt"
6.     Set MyTXT = MyFSO.OpenTextFile(filename, ForReading)
7.     Range("A3").Value = MyTXT.ReadLine
8.     Range("A4").Value = MyTXT.ReadLine
9.     Range("A5").Value = MyTXT.ReadLine
10.    Range("A6").Value = MyTXT.ReadLine
11.    Range("A7").Value = MyTXT.ReadLine
12.    MyTXT.Close
13.    Set MyFSO = Nothing
14.    Set MyTXT = Nothing
15. End Sub
```

■プロシージャの意味

1.「ラベルテキスト読込」プロシージャ開始
2. FileSystemObject型のオブジェクト変数「MyFSO」を使用することを宣言してインスタンスを生成
3. TextStream型のオブジェクト変数「MyTXT」を使用することを宣言
4. 文字列型の変数「filename」を使用することを宣言
5. 変数「filename」に実行中のプロシージャが記述されたブックが保存されているフォルダーの絶対パスと「¥9-3Practice.txt」を連結して代入
6. 変数「filename」のテキストファイルを読み込みモードで開いてオブジェクト変数「MyTXT」に代入
7. テキストファイルの1行目の文字列を読み込み、セル【A3】に入力
8. テキストファイルの2行目の文字列を読み込み、セル【A4】に入力
9. テキストファイルの3行目の文字列を読み込み、セル【A5】に入力
10. テキストファイルの4行目の文字列を読み込み、セル【A6】に入力
11. テキストファイルの5行目の文字列を読み込み、セル【A7】に入力
12. テキストファイルを閉じる
13. オブジェクト変数「MyFSO」の初期化
14. オブジェクト変数「MyTXT」の初期化
15. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

9-4 テキストファイルに書き込むには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「ラベルテキスト追加」プロシージャ

```
1. Sub ラベルテキスト追加()
2.     Dim MyFSO As New FileSystemObject
3.     Dim MyTXT As TextStream
4.     Dim filename As String
5.     filename = ThisWorkbook.Path & "¥9-4Practice.txt"
6.     Set MyTXT = MyFSO.OpenTextFile(filename, ForAppending)
7.     MyTXT.WriteBlankLines Lines:=3
8.     MyTXT.WriteLine Text:=Range("A9").Value
9.     MyTXT.WriteLine Text:=Range("A10").Value
10.    MyTXT.WriteLine Text:=Range("A11").Value
11.    MyTXT.WriteLine Text:=Range("A12").Value
12.    MyTXT.WriteLine Text:=Range("A13").Value
13.    MyTXT.Close
14.    Set MyFSO = Nothing
15.    Set MyTXT = Nothing
16. End Sub
```

■プロシージャの意味

1.「ラベルテキスト追加」プロシージャ開始
2. FileSystemObject型のオブジェクト変数「MyFSO」を使用することを宣言してインスタンスを生成
3. TextStream型のオブジェクト変数「MyTXT」を使用することを宣言
4. 文字列型の変数「filename」を使用することを宣言
5. 変数「filename」に実行中のプロシージャが記述されたブックが保存されているフォルダーの絶対パスと「¥9-4Practice.txt」を連結して代入
6. 変数「filename」のテキストファイルを書き込み（追記）モードで開いてオブジェクト変数「MyTXT」に代入
7. テキストファイルに3行改行を書き込む
8. テキストファイルにセル【A9】の値を書き込む
9. テキストファイルにセル【A10】の値を書き込む
10. テキストファイルにセル【A11】の値を書き込む
11. テキストファイルにセル【A12】の値を書き込む
12. テキストファイルにセル【A13】の値を書き込む
13. テキストファイルを閉じる
14. オブジェクト変数「MyFSO」の初期化
15. オブジェクト変数「MyTXT」の初期化
16. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

9-5 CSVファイルを読み込むには？

❶ 次のようにプロシージャを入力します。
※VBEを起動し、《挿入》→《標準モジュール》をクリックします。

■「お客様情報CSV読込」プロシージャ

```
Sub お客様情報CSV読込()
    Dim MyFSO As New FileSystemObject
    Dim MyTXT As TextStream
    Dim filename As String
    Dim customer As Variant
    filename = ThisWorkbook.Path & "¥9-5お客様情報.csv"
    Set MyTXT = MyFSO.OpenTextFile(filename, ForReading)
    Range("A4").Select
    MyTXT.SkipLine
    Do Until MyTXT.AtEndOfStream = True
        customer = Split(MyTXT.ReadLine, ",")
        With ActiveCell
            .Value = customer(0)
            .Offset(0, 1).Value = customer(1)
            .Offset(0, 2).Value = customer(2)
            .Offset(1, 0).Select
        End With
    Loop
    MyTXT.Close
    Set MyFSO = Nothing
    Set MyTXT = Nothing
End Sub
```

■プロシージャの意味

```
「お客様情報CSV読込」プロシージャ開始
    FileSystemObject型のオブジェクト変数「MyFSO」を使用することを宣言してインスタンスを生成
    TextStream型のオブジェクト変数「MyTXT」を使用することを宣言
    文字列型の変数「filename」を使用することを宣言
    バリアント型の変数「customer」を使用することを宣言
    変数「filename」に実行中のプロシージャに記述されたブックが保存されているフォルダーの絶対パスと「¥9-5お客様情報.csv」を連結して代入
    変数「filename」のテキストファイルを読み込みモードで開いてオブジェクト変数「MyTXT」に代入
    セル【A4】を選択
    テキストファイルの読み込み位置を1行分スキップ
    読み込み位置がテキストファイルの末尾になるまで処理を繰り返す
        1行分の文字列を読み込んで区切り文字「,」で分割し、変数「customer」に配列として代入
        アクティブセルの
            値に配列変数「customer(0)」の値を入力
            1列右のセルに配列変数「customer(1)」の値を入力
            2列右のセルに配列変数「customer(2)」の値を入力
            1行下のセルを選択
        Withステートメント終了
    10行目に戻る
    テキストファイルを閉じる
    オブジェクト変数「MyFSO」の初期化
    オブジェクト変数「MyTXT」の初期化
プロシージャ終了
```

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

9-6 CSVファイルに書き込むには？

❶「お客様情報CSV読込」プロシージャを実行します。

※「9-6お客様情報1.csv」ファイルから15件のお客様情報が読み込まれ、セル範囲【A4：C18】に入力されます。

❷ セル範囲【A19：C19】に、新しいお客様情報「50016　近藤　佐紀　120,000」を入力します。

❸ 次のようにプロシージャを入力します。

※VBEを起動し、《標準モジュール》→モジュール「Module1」を開いておきましょう。

■「お客様情報CSV書込」プロシージャ

```
Sub お客様情報CSV書込()
    Dim MyFSO As New FileSystemObject
    Dim MyTXT As TextStream
    Dim filename As String
    Dim customer(2) As Variant
    filename = ThisWorkbook.Path & "¥9-6お客様情報2.csv"
    Set MyTXT = MyFSO.OpenTextFile(filename, ForWriting, True)
    Range("A3").Select
    Do Until ActiveCell.Value = ""
        With ActiveCell
            customer(0) = .Value
            customer(1) = .Offset(0, 1).Value
            customer(2) = .Offset(0, 2).Value
            .Offset(1, 0).Select
        End With
        MyTXT.WriteLine Text:=Join(customer, ",")
    Loop
    MyTXT.Close
    Set MyFSO = Nothing
    Set MyTXT = Nothing
End Sub
```

■プロシージャの意味

1.「お客様情報CSV書込」プロシージャ開始
2.　FileSystemObject型のオブジェクト変数「MyFSO」を使用することを宣言してインスタンスを生成
3.　TextStream型のオブジェクト変数「MyTXT」を使用することを宣言
4.　文字列型の変数「filename」を使用することを宣言
5.　バリアント型の配列変数「customer」を3要素使用することを宣言
6.　変数「filename」に実行中のプロシージャが記述されたブックが保存されているフォルダーの絶対パスと「¥9-6お客様情報2.csv」を連結して代入
7.　変数「filename」のテキストファイルを書き込み（上書き）モードで開いて（ファイルが存在しない場合は新規作成）、オブジェクト変数「MyTXT」に代入
8.　セル【A4】を選択
9.　アクティブセルが空文字（「""」）になるまで処理を繰り返す
10.　　アクティブセルの
11.　　値を配列変数「customer(0)」に代入
12.　　1列右のセルの値を配列変数「customer(1)」に代入
13.　　2列右のセルの値を配列変数「customer(2)」に代入
14.　　1行下のセルを選択
15.　　Withステートメント終了
16.　　配列変数「customer」の各要素を区切り文字「,」で結合した文字列と改行を書き込む
17.　9行目に戻る
18.　テキストファイルを閉じる
19.　オブジェクト変数「MyFSO」の初期化
20.　オブジェクト変数「MyTXT」の初期化
21. プロシージャ終了

※コンパイルを実行し、上書き保存しておきましょう。
※プロシージャの動作を確認します。

よくわかる

Microsoft® Excel® マクロ／VBA 超実践トレーニング

Office 2021／2019／2016／Microsoft 365 対応

(FPT2406)

Practice 標準解答

2024年11月13日　初版発行

著作／制作：株式会社富士通ラーニングメディア