した場合、そのセルを基準とする相対的な位置関係で、引数のセル参照が表すRangeオブジェクトが取得される。

ここでは、まず「ActiveCell」プロパティで、アクティブセル（選択された入力対象のセル）を表すRangeオブジェクトを取得。その後に「.Range ("A1：B1")」とすることで、アクティブセルを基準（A1）とする相対的な「A1：B1」のセル範囲を表すRangeオブジェクトを取得できる。アクティブセルがE2セルの場合、取得されるのはこのE2セルを基準とした1行×2列のセル範囲、つまりE2～F2セルを表すRangeオブジェクトだ。このセル範囲を選択し、選択範囲に対して太字や配置を設定する（**図19**）。

```
ActiveCell.Range("A1:B1").Select
```

図19 ActiveCellでアクティブセルを表すRangeオブジェクトを取得。それにRangeを指定して、アクティブセル（ここではE2セル）を基準（A1）とする相対的なA1～B1セル、つまりここではE2～F2セルを表すRangeオブジェクトを取得し、選択している

```
ActiveCell.Offset(1, 0).Range("A1:B1").Select
```

図20 やはりアクティブセルを表すRangeオブジェクトの「Offset」で、その1行下のセルを取得。さらにそのセル（ここではE3セル）を基準（A1）とする相対的なA1～B1セル、つまりここではE3～F3セルを表すRangeオブジェクトを取得し、選択している

●マクロ「書式設定3」を修正する

```
Sub 書式設定4()
    With ActiveCell.Range("A1:B1")
        .Font.Bold = True
        .HorizontalAlignment = xlCenter
    End With
    ActiveCell.Range("A2:B2").NumberFormatLocal = _
        "¥#,##0_);[赤](¥#,##0)"
End Sub
```

図21 「書式設定3」のSubプロシージャをコピーしてコードウインドウの下側に貼り付け、プロシージャ名を「書式設定4」に変更。そのコードを修正して簡潔にした。やはり、セルを選択することなく書式だけを変更する形にしている

```
    ActiveCell.Range("A2:B2").NumberFormatLocal = _
        "¥#,##0_);[赤](¥#,##0)"
```

図22 アクティブセルの1行下で、1行×2列のセル範囲を操作したい場合、Offsetを使わなくても、Rangeの引数に「"A2：B2"」と指定すれば、同様のセル範囲が取得できる。なお、行末に「 _」を付けて改行すると、次の行もつながった1行のコードと見なされる

この操作で選択範囲は変化したが、アクティブセル（ここではE2セル）は変わっていない。Rangeオブジェクトを対象とした「Offset」プロパティでは、対象のセルから指定した行数と列数だけずらした位置のセルを表すRangeオブジェクトを取得できる。ここでは1行と0列、つまり1行下のE3セルを取得している。そのセルを基準とした「Range ("A1：B1")」で、やはり1行×2列のセル範囲を表すRangeオブジェクトを取得。そのE3～F3セルを選択し、選択範囲に対して表示形式を設定している（**図20**）。

この「書式設定3」のSubプロシージャも、前の例と同様に修正しよう（**図21**）。対象範囲の指定方法が変わっただけで、修正の方法は「書式設定1」を「書式設定2」に変更したときと同様だ。見出しのセルの書式を変える操作は、Withの対象を「ActiveCell.Range ("A1：B1")」にする。

一方、アクティブセルの1行下で1行×2列のセル範囲は、Offsetプロパティを使わなくても、Rangeプロパティだけで取得できる。ActiveCellを対象としたRangeプロパティの引数に「A2：B2」を指定すればよい（**図22**）。なお、ここでは「=」の後に「 _」（半角スペースとアンダースコア）を付けて改行している。これは「行継続文字」と呼ばれ、その次の行もつながった1行のコードと見なされる。行継続文字を付けて改行した場合、前からつながっている行は1段階インデントするのが、一般的な作法だ。