

第十五章 四個 VBA 程式設計範例

承續第十四章，在本章中，我們將透過 4 個範例來學習 VBA。同時，在本章範例中將出現一些前面章節未介紹到的函數和語法，您也將能在範例的練習過程中體會。在本章中要練習的範例有：

●繪製螺栓

●繪製銷

●立體 Flange

●結合資料庫應用



15-1 前言

在實作本節章前，我們要對 VBA 裡的程式組織架構先做一複習式的說明。對 VBA 來說，在其編輯視窗中，您可以清楚的看到一個 VBA 程式可以有：程序、表單與模組等結構。

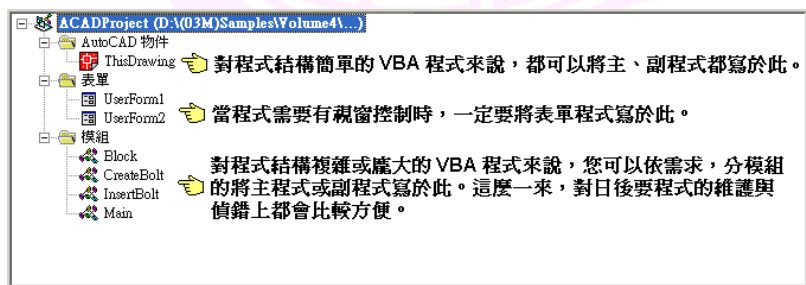


圖 15-1 VBA 程式的結構

對本章的範例來說，都分別採用了上述各種結構撰寫方式，而不同的結構除了本身的意義不同之外，也會影響該程式欲執行時所呼叫的指令名稱。這部分的

詳細資訊，請參考第十六章。

15-2 繪製螺栓

● 本範例完成圖形

我們這個範例的作法是很高級的。在機械專業裡，螺栓是最常被畫到的零件圖形之一。但是如果要將標準螺栓都以圖塊的方式來做，不要說螺栓內的尺寸變數多，就算要一個一個的將之畫成「圖塊」圖形，也要花上不少時間，而且不一定有效率。因此，聰明的手段就是以一個程式來一次搞定這類的標準螺栓圖形。那麼本程式特別之處在哪裡呢？如下所述：

1. 因為螺栓的設計條件參數是很多的，要一個一個都要使用者輸入，那一定會讓使用者煩死。再說，螺栓的設計是根據機械工作法規中所規定，一般的機械設計也都會因要節省製造成本而偏向儘量使用標準的螺栓，而不會搞怪的去設計規格特殊的螺栓。在這樣的專業背景下，將法規輸入程式中，就等於替使用者到機械工作法規中「查表」，人家一定會說您內行並歡迎這種程式的。
2. 既然「輸入規格就可自動查表」是本程式的第一項特色，那麼簡單的輸入條件應該就是這個程式的第二特色。本程式只要您回答四個問題，就可以畫出一個符合專業規格的螺栓。
3. 本程式還將是類似機械零件的「樣板」程式。只要是機械工作法規裡的其他零件（如標準齒輪、凸輪、鏈條..等），都可以模仿這樣的程式架構來建立，只是得到的圖形不同而已。「標準樣板」程式就是這個程式的第三項特色。

本範例將畫出一個如圖 15-2 所示的標準M3 系列螺栓：



圖 15-2 本程式的螺栓完成圖

本例將主要用到 `m`、`distance`、`centerPt` 和以螺栓編號為名的程序。以下部分我們將詳細來講述它們的功能以及程式設計的技巧。

- 本範例檔案：screw.dvb（本電子書範例光碟(03M)Samples\Volume4 目錄）

- 程序 m（ThisDrawing）

作用：繪製不同大小的標準螺栓。

(1)Public Sub m(d1, h, c, d, r, k, s, c2, dw As Double)

說明：定義畫螺栓的程序，其中 d1、h、c、d、r、k、s、c2、dw 等均為螺栓的設計變數。

(2)Dim sysOSMODE As Integer

說明：用 sysOSMODE 變數來儲存 AutoCAD 的鎖點模式。

(3)sysOSMODE = ThisDrawing.GetVariable("osmode")

說明：將 AutoCAD 中的鎖點模式存入 sysOSMODE 變數中，GetVariable 方法將用於設定 AutoCAD 系統變數。

(4)thisDrawing.SetVariable "osmode", 0

說明：設定鎖點模式為 0，即關閉 AutoCAD 的鎖點模式。這樣做的目的是要避免在畫圖的過程中取點發生偏差。請注意這點很重要：在您又需要鎖點時，就可以再利用儲存在 sysOSMODE 中的原模式來恢復。SetVariable 方法是用來取得 AutoCAD 的系統變數。

(5)thisDrawing.Utility.InitializeUserInput 32

說明：這條程式是用來初始 GetKeyword 的方法，設定為 32 是要讓使用者在螢幕上取點時，用虛線拉出橡皮拉線或方框。Utility 物件可用來提供一些實用功能，如：輸入點、輸入數值或獲得圖素影像等。

(6)On Error Resume Next

說明：這條程式是用來說明當一個執行時發生錯誤時，控制項將轉移到發生此錯誤語法後的語句，並於此再繼續執行。

(7)Dim fp, sp As Variant

說明：設定 fp 和 sp 變數來分別儲存螺栓前視圖的中心點和螺栓側視圖的基準點。注意：點可以用變式（variant）變數儲存。當點輸入後，以 fp 為例，fp(0) 用來儲存 fp 點的 x 值，fp(1) 則儲存 fp 點的 y 值，fp(2) 則儲存 fp 點的 z 值。

(8)fp = ThisDrawing.Utility.GetPoint(, "請輸入螺栓前視圖的中心點:")

(9)sp = ThisDrawing.Utility.GetPoint(fp, "請輸入螺栓側視圖的基準點:")

(10)l = ThisDrawing.Utility.GetDistance(, "請輸入螺栓長度 L:")

(11)rotsita = ThisDrawing.Utility.GetAngle(, "請輸入螺栓的旋轉角度<0>:")

說明：利用 Utility 的方法，將插入圖形的位置參數儲存到變數 fp、sp、l、rotsita 中。

(12)If Err Then

(13) rotsita = 0#

(14) Err.Clear

(15)End If

說明：一個好程式起體質就應該強壯，意即在提示輸入時如果有操作者有失誤操作的情況發生，就不應該因產生錯誤結果而停止或忽略錯誤結果而繼續執行。如果在以上提示輸入角度的語法中使用者不熟悉 AutoCAD，不知如何輸入角度甚至按下 <Esc> 鍵，則 Err 將被設定為一個非零的數，對應的角度變數 rostita 就被設為 0。"#" 代表此數為浮點數。

(16)If s >= 1 Then

(17) s = 1 - 2# * k

(18)End If

說明：設定 s 值。

(19)hrad = d / 2#

說明：設定半徑值 hrad。

(20)Dim mx As String

(21)mx = "M" & CStr(Fix(d1)) & CStr(Fix(h)) & CStr(Fix(l))

說明：用 mx 來儲存螺栓圖塊的名稱，規則是以 "M" 開頭加上圖形參數 d1、h 和 l 的整數值。Fix 函數是取浮點數的整數部分，CStr 函數則可強制將一個運算式轉換成字元資料類型。

(22)Dim circ As AcadCircle

(23)Set circ = ThisDrawing.ModelSpace.AddCircle(fp, hrad)

說明：現在開始先畫前視圖中的圓，ModelSpace 表示在模型空間中操作，本程式作用是要以 fp 點為中心，以 hrad 為半徑畫圓。

(24)Dim fpstr As String

(25)fpstr = fp(0) & "," & fp(1) & "," & fp(2)

(26)ThisDrawing.SendCommand ("_polygon" & vbCr & "6" & vbCr & fpstr & vbCr & "c" & vbCr & hrad & vbCr)

說明：現在要使用 SendCommand 方法來向 AutoCAD 發出指令。這就類似 LISP 裡的 Command。本程式的作用是畫一個正多邊形。其中，vbCr 代表 <Enter>，fpstr 則是表示中心點的字串。

(27)Dim flagno As Integer

(28)flagno = 0

說明：使用 flagno 來作為一個判斷值。初始值設為 0。

(29)Dim iblock As Integer

(30)iblock = ThisDrawing.Blocks.Count

說明：將目前圖形中的圖塊的數量存入變數 iblock 中。

(31)While (iblock > 0)

(32) If ThisDrawing.Blocks.Item(iblock - 1).Name = mx Then

(33) flagno = 1

(34) End If

(35) iblock = iblock - 1

(36)Wend

說明：判斷目前圖中圖塊的數量是否大於 0，即是否有圖塊存在。如果存在圖塊，那麼就遍歷所有圖塊，並判斷是否有 mx 變數表示的圖塊存在，若有，設 flagno 為 1。這樣設計的目的是：可以充分利用存在的圖塊，而不用在去重新產生，以節省圖檔的容量與畫出時間。

```
(37)If flagno = 0 Then
(38)    ax0 = sp(0)
(39)    ax1 = ax0 - h
(40)    ax2 = ((c / 2# - d / 2#) / 1.732 + ax0) - h
(41)    ax3 = (1.5 - 1.414) * d1 + ax0 - h
(42)    ax5 = ax0 + r + c2
(43)    ax6 = ax0 - s + l - d1 / 5# + c2
(44)    ax7 = ax0 + l - s + c2
(45)    ax8 = ax0 + l - k + c2
(46)    ax9 = ax0 + l + c2
(47)    ax10 = (ax1 + ax2) / 2#
(48)    ax11 = ax8 + d1 / 10#
(49)    ay0 = sp(1)
(50)    ay2 = ay0 + d1 / 2#
(51)    ay3 = ay0 + c * 3 / 8
(52)    ay4 = ay0 + d / 2#
(53)    ay5 = ay0 + c / 2#
(54)    ay6 = ay0 + r + d1 / 2#
(55)    ay7 = ay0 + d1 / 2# - k
(56)    ay8 = (ay4 + ay5) / 2#
(57)    ay9 = ay2 - d1 / 10#
(58)ccrad = 1.5 * d1
```

說明：計算出圖形所需各點的 x 與 y 座標值。

```
(59)Dim p10, p32, p13, p14, p25, p05, p06, p52, p72, p77, p70, p82,
    p80, p90, p97, p62, p02, p108, p79, p119, p1, p2 As Variant
(60)Dim utilObj As Object
(61)Set utilObj = ThisDrawing.Utility
(62)    utilObj.CreateTypedArray p10, vbDouble, ax1, ay0, 0
(63)    utilObj.CreateTypedArray p32, vbDouble, ax3, ay2, 0
(64)    utilObj.CreateTypedArray p13, vbDouble, ax1, ay3, 0
(65)    utilObj.CreateTypedArray p14, vbDouble, ax1, ay4, 0
```



```

(66)    utilObj.CreateTypedArray p25, vbDouble, ax2, ay5, 0
(67)    utilObj.CreateTypedArray p05, vbDouble, ax0, ay5, 0
(68)    utilObj.CreateTypedArray p06, vbDouble, ax0 + c2, ay6, 0
(69)    utilObj.CreateTypedArray p52, vbDouble, ax5, ay2, 0
(70)    utilObj.CreateTypedArray p72, vbDouble, ax7, ay2, 0
(71)    utilObj.CreateTypedArray p77, vbDouble, ax7, ay7, 0
(72)    utilObj.CreateTypedArray p70, vbDouble, ax7, ay0, 0
(73)    utilObj.CreateTypedArray p82, vbDouble, ax8, ay2, 0
(74)    utilObj.CreateTypedArray p80, vbDouble, ax8, ay0, 0
(75)    utilObj.CreateTypedArray p90, vbDouble, ax9, ay0, 0
(76)    utilObj.CreateTypedArray p97, vbDouble, ax9, ay7, 0
(77)    utilObj.CreateTypedArray p62, vbDouble, ax6, ay2, 0
(78)    utilObj.CreateTypedArray p02, vbDouble, ax0, ay2, 0
(79)    utilObj.CreateTypedArray p108, vbDouble, ax10, ay8, 0
(80)    utilObj.CreateTypedArray p79, vbDouble, ax7, ay9, 0
(81)    utilObj.CreateTypedArray p119, vbDouble, ax11, ay9, 0
(82)    utilObj.CreateTypedArray p1, vbDouble, ax0 + c2, ay0, 0
(83)    utilObj.CreateTypedArray p2, vbDouble, ax0 + c2, ay0 + dw / 2, 0

```

說明：設定一些確定圖形所需的點。其中 CreateTypedArray 將用於建立一個包含某種類型變數陣列的變式變數。如："utilObj.CreateTypedArray p2, vbDouble, ax0 + c2, ay0 + dw / 2, 0" 就表示要建立一個點 p2。x、y、z 的值均為雙精度浮點型 (vbDouble)。p2(0)、p2(1) 和 p2(2) 的值分別為：ax0 + c2、ay0 + dw / 2 和 0。

(84)'建立圖塊

```
(85)Set blockObj = ThisDrawing.Blocks.Add(sp, mx)
```

說明：使用 mx 中的字串為圖塊名稱，以 sp 點為基點建立圖塊。

```

(86)Dim linea, lineb, linec, lined, linee, linef, lineg, lineh, linei,
    linej, linek, linep,

```

```
(87)lineq As AcadLine
```

```

(88)    Set linea = blockObj.AddLine(p10, p14)
(89)    Set lineb = blockObj.AddLine(p14, p25)
(90)    Set linec = blockObj.AddLine(p25, p05)
(91)    Set lined = blockObj.AddLine(p05, sp)
(92)    'Set linee = blockObj.AddLine(p06, sp)

```

```
(93) Set linef = blockObj.AddLine(p32, p02)
(94) Set lineg = blockObj.AddLine(p52, p82)
(95) Set lineh = blockObj.AddLine(p82, p80)
(96) Set linei = blockObj.AddLine(p82, p97)
(97) Set linej = blockObj.AddLine(p97, p90)
(98) Set linek = blockObj.AddLine(p72, p70)
(99) Set linep = blockObj.AddLine(p1, p2)
(100) Set lineq = blockObj.AddLine(p2, p05)
```

說明：在圖塊中加入一些線。

```
(101)dist = distance(p32, p10)
```

說明：呼叫 distance 程序以取得 p32 與 p10 兩點之間的距離，並將之存入 dist 變數中。

```
(102)pi = 3.1415926536
```

說明：與 Visual LISP 中不同，在 VBA 裡，pi 需要賦予一值。

```
(103)ang1 = Atn(Sqr(ccrad ^ 2 - (dist / 2#) ^ 2) * 2 / dist)
(104) ang2 = ThisDrawing.Utility.AngleFromXAxis(p32, p10)
(105) ang3 = ang1 + ang2 - 2 * pi
(106) Dim cenPt As Variant
(107) cenPt = ThisDrawing.Utility.PolarPoint(p32, ang3, ccrad)
(108) ang4 = ThisDrawing.Utility.AngleFromXAxis(cenPt, p32)
(109) ang5 = ThisDrawing.Utility.AngleFromXAxis(cenPt, p10)
(110) Dim arca As AcadArc
(111) Set arca = blockObj.AddArc(cenPt, ccrad, ang4, ang5)
```

說明：計算圓弧中心點、半徑、起始角與終止角，以在圖塊中加入弧段。
其中，PolarPoint 用於求某點的指定角度和指定距離處的點，並傳回該點；
AngleFromXAxis 將用於求某條線與 X 軸的夾角。

```
(112)Dim arcb As AcadArc
(113)Dim cenPt2 As Variant
(114)'用 p108，p13，p32 三點畫弧
```



```
(115)cenPt2 = centerPt(p108, p13, p32)
(116)arcb_ra = distance(cenPt2, p108)
(117)arcb_angs = ThisDrawing.Utility.AngleFromXAxis(cenPt2, p108)
(118)arcb_ange = ThisDrawing.Utility.AngleFromXAxis(cenPt2, p32)
(119)Set arcb = blockObj.AddArc(cenPt2, arcb_ra, arcb_angs, arcb_ange)
```

說明：在圖塊中插入第二條弧。

```
(120)distc = distance(p06, p52)
(121)ang1c = Atn(Sqr(r ^ 2 - (distc / 2#) ^ 2) * 2 / distc)
(122)ang2c = ThisDrawing.Utility.AngleFromXAxis(p06, p52)
(123)ang3c = ang1c + ang2c - 2 * pi
(124)Dim cenPt3 As Variant
(125)cenPt3 = ThisDrawing.Utility.PolarPoint(p06, ang3c, r)
(126)arcc_angs = ThisDrawing.Utility.AngleFromXAxis(cenPt3, p06)
(127)arcc_ange = ThisDrawing.Utility.AngleFromXAxis(cenPt3, p52)
(128)Dim arcc As AcadArc
(129)Set arcc = blockObj.AddArc(cenPt3, r, arcc_angs, arcc_ange)
```

說明：在圖塊中插入第三條弧。

```
(130)Dim linel, linem As AcadLine
(131) Set linel = blockObj.AddLine(p62, p79)
(132)Set linem = blockObj.AddLine(p79, p119)
```

說明：在圖塊中插入兩條線。

```
(133)linel.Color = 3
(134) linem.Color = acGreen
```

說明：於此設定這兩條線的顏色為綠色，3 代表顏色為綠色。另一種寫法是用較容易記憶的常數代替，如：acByBlock(0)，acRed(1)，acYellow(2)，acGreen(3)，acCyan(4)，acBlue(5)，acMagenta(6) 與 acWhite(7) 等。

```
(135)Dim acadent As AcadEntity
(136)For Each acadent In blockObj
(137) acadent.Mirror p10, sp
(138)Next acadent
```

說明：遍歷圖塊中物件，將所有物件均以 p10 和 sp 為端點的線鏡像。

(139)End If

說明：結束 if 語法，本句的意思是：若圖塊已存在就直接插入，若不存在則執行前述的語法來畫出一螺栓圖塊。

(140)Dim insertPt As Variant

(141)utilObj.CreateTypedArray insertPt, vbDouble, sp(0), sp(1), sp(2)

說明：設定圖塊的插入點 insertPt。

(142)Dim blockRefObj As AcadBlockReference

(143)Set blockRefObj = ThisDrawing.ModelSpace.InsertBlock(insertPt, mx, 1#,1#, 1#, 0)

說明：定義圖塊參考，圖塊參考的作用是用來顯示圖塊定義中所儲存的資料。InsertBlock 方法在 insertPt 點的插入名稱爲 mx，x 方向，y 方向與 z 方向的插入比例均爲 1，旋轉角爲 0。

(144)ThisDrawing.Regen acActiveViewport

說明：重新計算所有物件的螢幕座標，並重新產生整個圖形。它還會重新建立圖形資料庫的索引，從而讓顯示與物件選擇功能更好。

(145)ThisDrawing.SetVariable "osmode", sysOSMODE

說明：程式執行結束後，恢復原來的 AutoCAD 鎖點模式。

(146)End Sub

● 程序 distance (ThisDrawing)

作用：傳回兩點之間的距離。

(147)Function distance(sp As Variant, ep As Variant) As Double

(148) Dim x As Double

```

(149) Dim y As Double
(150) Dim z As Double
(151) x = sp(0) - ep(0)
(152) y = sp(1) - ep(1)
(153) distance = Sqr((x ^ 2) + (y ^ 2))
(154)End Function

```

說明：請注意：作為參數傳遞的兩點是變式變數。在 VBA 中沒有如 LISP 般可直接擷取兩點距離的函數 distance，因此，編寫此實用的程序。

● 程序 centerPt

作用：傳回三點確定圓的圓心。原理與第四章所述相同，求三點中任意兩條直線的中垂線交點。

```

(155)Function centerPt(pt1, pt2, pt3 As Variant) As Variant
(156)Dim line1, line2, line3, line4 As AcadLine

```

作用：定義 4 條直線輔助以求取中心點。

```

(157)Set line1 = ThisDrawing.ModelSpace.AddLine(pt1, pt2)
(158)Set line2 = ThisDrawing.ModelSpace.AddLine(pt2, pt3)
(159)Dim lcenPt1, lcenPt2 As Variant
(160)Dim util As Object
(161)Set util = ThisDrawing.Utility
(162)util.CreateTypedArray lcenPt1, vbDouble, (pt1(0) + pt2(0)) / 2, (pt1(1) + pt2(1)) / 2, 0
(163)util.CreateTypedArray lcenPt2, vbDouble, (pt2(0) + pt3(0)) / 2, (pt2(1) + pt3(1)) / 2, 0
(164)Dim angle1, angle2 As Double
(165)angle1 = ThisDrawing.Utility.AngleFromXAxis(pt1, pt2)
(166)angle2 = ThisDrawing.Utility.AngleFromXAxis(pt2, pt3)
(167)Set line3 = ThisDrawing.ModelSpace.AddLine(lcenPt1,
(168)ThisDrawing.Utility.PolarPoint(lcenPt1, angle1 + 3.1415926536 / 2, 100))
(169)Set line4 = ThisDrawing.ModelSpace.AddLine(lcenPt2,
(170)ThisDrawing.Utility.PolarPoint(lcenPt2, angle2 + 3.1415926536 / 2, 100))

```

說明：畫出兩條中垂線 line3 和 line4。

```

(171)centerPt = line4.IntersectWith(line3, acExtendBoth)

```

說明：使用 IntersectWith 方法來求出 line3 和 line4 交點。acExtendBoth 表示兩條直線若無交點則求出其延長線的交點。

(172)line1.Delete

(173)line2.Delete

(174)line3.Delete

(175)line4.Delete

說明：圓心求出後，將那 4 條輔助直線刪除。

(176)End Function

- 繪螺栓的程序群(ThisDrawing，將螺栓的資料庫均以程序參數的方式來儲存)

作用：將M系列的各種螺栓設計條件參數傳遞給程序 m，以減少提示文句的詢問。

Public Sub m3()

Call m(3#, 2#, 6.4, 5.3, 0.2, 0.6, 12#, 0.4, 4.6)

End Sub

Public Sub m4()

Call m(4#, 2.8, 8.1, 6.8, 0.3, 0.8, 14#, 0.4, 5.9)

End Sub

Public Sub m5()

Call m(5#, 3.5, 9.2, 7.8, 0.3, 0.9, 16#, 0.5, 6.9)

End Sub

Public Sub m6()

Call m(6#, 4#, 11.5, 9.8, 0.5, 1#, 18#, 0.5, 8.9)

End Sub

Public Sub m8()

Call m(8#, 5.5, 15#, 12.6, 0.5, 1.2, 22#, 0.6, 11.6)

End Sub

Public Sub m10()

Call m(10#, 7#, 19.6, 16.5, 0.8, 1.5, 26#, 0.6, 14.6)

End Sub

Public Sub m12()

Call m(12#, 8#, 21.9, 18#, 0.8, 2#, 30#, 0.6, 16.6)

End Sub

Public Sub m14()

Call m(14#, 9#, 25.4, 21#, 0.8, 2#, 34#, 0.6, 19.6)

End Sub

Public Sub m16()

Call m(16#, 10#, 27.7, 23#, 1.2, 2#, 38#, 0.8, 22.5)

End Sub

Public Sub m18()

Call m(18#, 12#, 31.2, 26#, 1.2, 2.5, 42#, 0.8, 25.3)

End Sub

Public Sub m20()

Call m(20#, 13#, 34.6, 29#, 1.2, 2.5, 46#, 0.8, 28.2)

End Sub

Public Sub m22()

Call m(22#, 14#, 37#, 31#, 1.2, 2.5, 50#, 0.8, 31.7)

End Sub

Public Sub m24()

Call m(24#, 15#, 41.6, 34#, 1.6, 3#, 54#, 0.8, 33.6)

End Sub

Public Sub m27()

Call m(27#, 17#, 47.3, 39#, 1.6, 3#, 60#, 0.8, 38)

End Sub

Public Sub m30()

Call m(30#, 19#, 53.1, 44#, 1.6, 3.5, 66#, 0.8, 42.7)

End Sub

Public Sub m36()

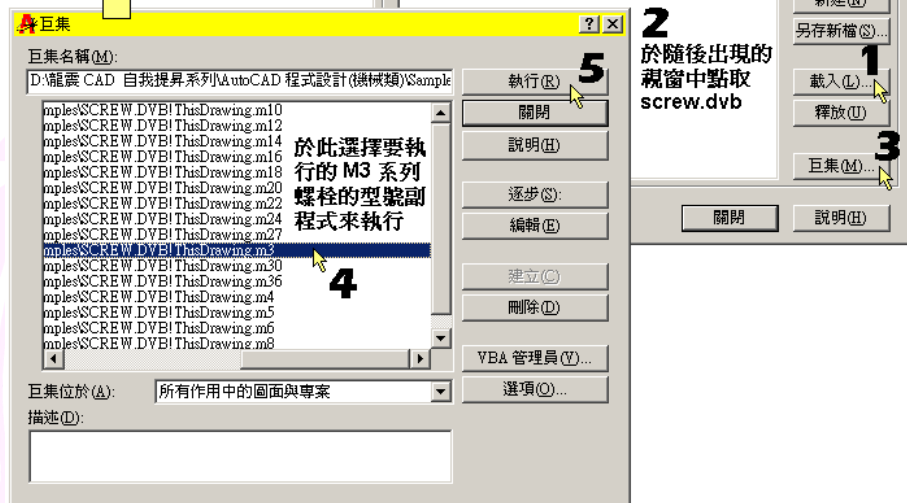
Call m(36#, 23#, 63.5, 53#, 2#, 4#, 78#, 0.8, 51.1)

End Sub

● 本範例的執行步驟

請輸入螺栓前視圖的中心點：
請輸入螺栓側視圖的基準點：
請輸入螺栓長度 L:50 <Enter>
請輸入螺栓的旋轉角度<0>: <Enter>

6



注意：從此圖的執行看來，這個程式的執行方法是很笨拙的。因為大家都知道：

M3 系列的螺栓是很常用的，型號是耳熟能詳的，但是以這樣的方式來執行，程式設計的介面親和力方面的分數將被打「零分」！那怎麼辦？別急！我們有二個方法來處理這個問題：

1. 首先，我們可以利用下拉式功能表點取的方法來解決這個問題（第十六章中介紹）
2. 修改此程式，讓其執行後能先出現一個選型號的介面（不過，我們將這個方法當本章習題，請您先動動腦，然後我們再於習題解答中揭曉答案！）

圖 15-3 執行 screw.dvb

15-3 繪製銷

● 本範例完成圖形

畫銷的程式是由一個表單和一個模組所組成。本範例最大的特點就是：在 VBA 中使用了 FSO 功能來處理檔案物件及文字流。本範例將畫出一個如圖

15-4 所示的標準銷：

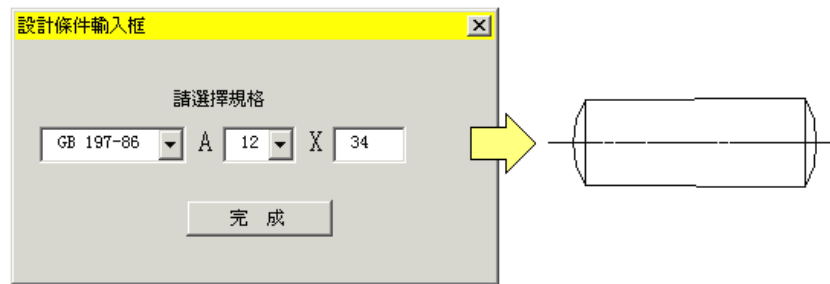


圖 15-4 本程式的銷完成圖

本例主要將用到將由 keys 模組來執行 UserForm1 表單而成。而所需執行的副程式內容也一併寫在表單的語法中，如：commandButton1_Click()、UserForm_Initialize()、getdata() 與 stoa() 等。以下部分，我們將詳細來講述它們的功能與程式設計的技巧。

- 本範例檔案：

- (1) Pin(AutoCAD 2000).DVB(本電子書範例光碟(03M)Samples\Volume4 目錄)
- (2) Pin(AutoCAD 2008).DVB(本電子書範例光碟(03M)Samples\Volume4 目錄)

注意：以上兩檔案，當您已決定您要使用的 AutoCAD 版本後，請將其中之一的檔案更名為 **pin.dvb**。

- (3) gb117.txt (本電子書範例光碟(03M)Samples\Volume4 目錄)
- (4) gb119.txt (本電子書範例光碟(03M)Samples\Volume4 目錄)

- 模組：keys

- (1)Public Sub show()

說明：定義 show 模組程序。

- (2)UserForm1.show

說明：執行 UserForm1 表單。

- (3)End Sub

- 表單：UserForm1

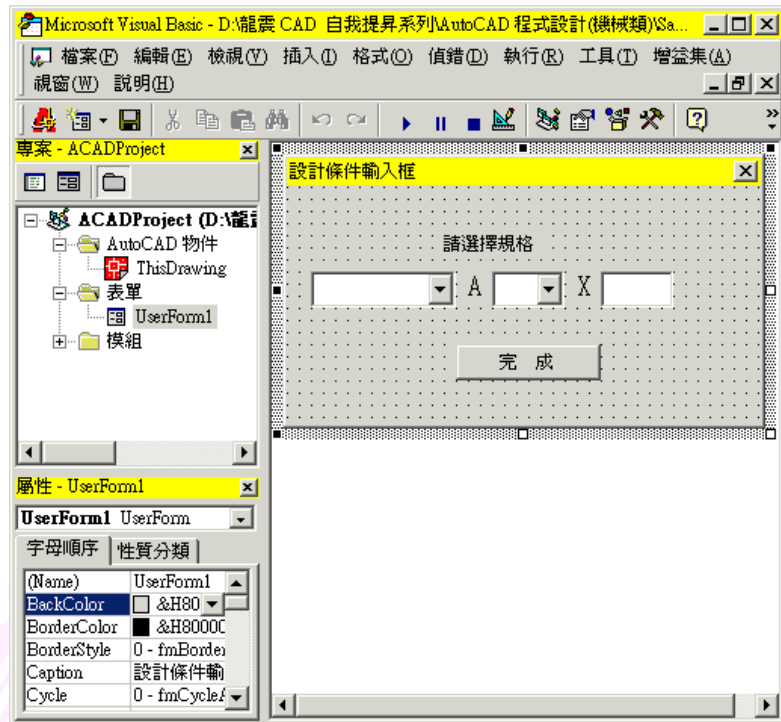


圖 15-5 本範例的輸入框表單

表單 UserForm1 的程式碼原文：

(1)Dim sa(9) As Double

說明：定義一個整體變數 sa 陣列來用於傳遞參數。

(2)Private Sub CommandButton1_Click()

說明：當使用者按下「完成」鈕時執行以下程式：

(3)Dim l_up As Double

(4)Dim l_low As Double

說明：用來定義銷長度的最大值和最小值。

(5)Dim a As Double

(6)Dim c As Double

(7)Dim d As Double

說明：定義銷的一些參數。

(8)i = ComboBox1.ListIndex

(9)j = ComboBox2.ListIndex

說明：擷取下拉式顯示盒中的順序，並分別儲存於 i 和 j 變數中。其中，i 代表是哪一種類型的銷。

(10)fn = "d:/(03M)Samples/Volume4/gb119.txt"

說明：設定欲開啓的檔案名稱。

(11)n = 0

(12)key = CInt(ComboBox2.Text)

(13)Call getdata(fn, key, n)

說明：呼叫 getdata 程序來擷取銷的參數。

(14)d = sa(0)

(15)a = sa(1)

(16)c = sa(2)

(17)l_low = sa(3)

(18)l_up = sa(4)

(19)If i = 1 Then

(20) fn = " d:/(03M)Samples/Volume4/gb117.txt"

(21) n = 0

(22) key = CInt(ComboBox2.Text)

(23) Call getdata(fn, key, n)

(24) d = sa(0)

(25) a = sa(1)

(26) l_low = sa(2)

(27) l_up = sa(3)

(28)End If

說明：根據 i 的擷取值（即看是哪一種銷）來設定參數。

(29)l = Val(TextBox1.Text)

(30)While ((l < l_low) Or (l > l_up))

(31) l = InputBox("銷的長度必須在" & l_low & " 和 " & l_up & " 之間，請

再輸入一次！")

(32)Wend

說明：判斷如果輸入的銷的長度不合法，則提示重新輸入。

(33)UserForm1.Hide

說明：隱藏交談框。

(34)Dim sysOSMODE As Integer

(35)sysOSMODE = ThisDrawing.GetVariable("osmode")

(36)ThisDrawing.SetVariable "osmode", 0

說明：儲存並設定系統變數。

(37)Dim p1, p2, p3, p4, p5, p6, p7, p8, p9, insertPt As Variant

(38)Dim utilObj As Object

(39)Set utilObj = ThisDrawing.Utility

說明：定義一些參數。

(40)If i = 0 Then

說明：如果是 GB199 的銷，則執行以下代碼：

(41)d = Val(ComboBox2.Text)

(42)l = Val(TextBox1.Text)

(43)a = (1 - 3 ^ 0.5 / 2) * d

(44)insertPt = ThisDrawing.Utility.GetPoint("請輸入插入點:")

(45)utilObj.CreateTypedArray p1, vbDouble, insertPt(0), insertPt(1), 0

(46)utilObj.CreateTypedArray p2, vbDouble, insertPt(0), insertPt(1) + d / 2 - Tan(15 * 3.141592654 / 180) * c, 0

(47)utilObj.CreateTypedArray p3, vbDouble, insertPt(0) + c, insertPt(1), 0

(48)utilObj.CreateTypedArray p4, vbDouble, insertPt(0) + c, insertPt(1) + d / 2, 0

(49)utilObj.CreateTypedArray p5, vbDouble, insertPt(0) + l - a, insertPt(1), 0

(50)utilObj.CreateTypedArray p6, vbDouble, insertPt(0) + l - a, insertPt(1) + d / 2, 0

(51)utilObj.CreateTypedArray p7, vbDouble, insertPt(0) + l - d, insertPt(1), 0

(52)utilObj.CreateTypedArray p8, vbDouble, insertPt(0) - l / 10, insertPt(1), 0

(53)utilObj.CreateTypedArray p9, vbDouble, insertPt(0) + 1 + 1 / 10, insertPt(1), 0

說明：計算並設定畫銷所需要的點。

(54)Dim mx As String

(55)mx = ComboBox1.Text & "A" & ComboBox2.Text & TextBox1.Text

說明：使用 mx 變數來儲存圖塊名稱。

(56)Dim flagno As Integer

(57)flagno = 0

說明：使用 flagno 來判斷圖塊是否已經存在。

(58)Dim iblock As Integer

(59)iblock = ThisDrawing.Blocks.Count

(60)While (iblock > 0)

(61) If ThisDrawing.Blocks.Item(iblock - 1).Name = mx Then

(62) flagno = 1

(63) End If

(64) iblock = iblock - 1

(65)Wend

說明：檢查圖塊是否已存在，再根據存在與否來設定 flagno 值。

(66)If flagno = 0 Then

說明：如果圖塊不存在則建立一個新的。

(67)' Create the block

(68)Set blockObj = ThisDrawing.Blocks.Add(insertPt, mx)

(69)Dim linea, lineb, linec, lined, linee, linef As AcadLine

(70)Dim arc As AcadArc

(71)Set linea = blockObj.AddLine(p1, p2)

(72)Set lineb = blockObj.AddLine(p3, p4)

(73)Set linec = blockObj.AddLine(p5, p6)

(74)Set lined = blockObj.AddLine(p2, p4)

(75)Set linee = blockObj.AddLine(p4, p6)

(76)Set arc = blockObj.AddArc(p7, d, 0, 3.141592654 / 6)

說明：繪直線和弧構成銷。

(77)Dim acadent As AcadEntity

(78)For Each acadent In blockObj

(79) acadent.Mirror p1, p3

(80)Next acadent

說明：鏡像“半”個銷。

(81)Set linef = blockObj.AddLine(p8, p9)

(82)Dim linetypeName As String

(83)linetypeName = "CENTER"

說明：從 acad.lin 線型檔案中載入 "CENTER" 線型。

(84)On Error Resume Next

(85)'鎖點錯誤

(86) ThisDrawing.Linetypes.Load linetypeName, "acad.lin"

(87) linef.Linetype = linetypeName

說明：畫出中心線並設定線型。

(88)End If

(89)Dim blockRefObj As AcadBlockReference

(90)Set blockRefObj = ThisDrawing.ModelSpace.InsertBlock(insertPt, mx, 1#, 1#, 1#, 0)

說明：根據插入點來插入圖塊。

(91)ThisDrawing.Regen acActiveViewport

說明：重新產生圖形。

(92)Else

說明：如果是 GB197 的銷，則執行以下程式（以下過程與前述 GB199 相

同)：

```
(93)""""GB197
(94)d = Val(ComboBox2.Text)
(95)l = Val(TextBox1.Text)
(96)a = (1 - 3 ^ 0.5 / 2) * d
(97)R2 = d + (1 - 2 * a) / 50
(98)b = (R2 ^ 2 - (R2 - a) ^ 2) ^ 0.5
(99)insertPt = ThisDrawing.Utility.GetPoint("請輸入插入點:")
(100)utilObj.CreateTypedArray p1, vbDouble, insertPt(0) + a, insertPt(1), 0
(111)utilObj.CreateTypedArray p2, vbDouble, insertPt(0) + a, insertPt(1) + d / 2, 0
(112)utilObj.CreateTypedArray p3, vbDouble, insertPt(0) + l - a, insertPt(1), 0
(113)utilObj.CreateTypedArray p4, vbDouble, insertPt(0) + l - a, insertPt(1) + b, 0
(114)utilObj.CreateTypedArray p5, vbDouble, insertPt(0) + d, insertPt(1), 0
(115)utilObj.CreateTypedArray p6, vbDouble, insertPt(0) + l - R2, insertPt(1), 0
(116)utilObj.CreateTypedArray p7, vbDouble, insertPt(0) - l / 10, insertPt(1), 0
(117)utilObj.CreateTypedArray p8, vbDouble, insertPt(0) + l + l / 10, insertPt(1), 0
(118)    mx = ComboBox1.Text & "A" & ComboBox2.Text & TextBox1.Text
(119)flagno = 0
(120)iblock = ThisDrawing.Blocks.Count
(121)While (iblock > 0)
(122)    If ThisDrawing.Blocks.Item(iblock - 1).Name = mx Then
(123)        flagno = 1
(124)    End If
(125)    iblock = iblock - 1
(126)Wend
(127)If flagno = 0 Then
(128)    '建立圖塊
(129)    Set blockObj = ThisDrawing.Blocks.Add(insertPt, mx)
(130)    ' Dim linea, lineb, linec, lined, linee, linef As AcadLine
(131)    Dim arc1, arc2 As AcadArc
(132)    Set linea = blockObj.AddLine(p1, p2)
(133)    Set lineb = blockObj.AddLine(p3, p4)
(134)    Set lined = blockObj.AddLine(p2, p4)
(135)    Set arc1 = blockObj.AddArc(p5, d, 3.141592654 * 5 / 6, 3.141592654)
(136)    Set arc2 = blockObj.AddArc(p6, R2, 0, Atn(b / (R2 - a)))
(137)    For Each acadent In blockObj
(138)        acadent.Mirror p1, p3
```

```

(139) Next acadent
(140) Set linef = blockObj.AddLine(p7, p8)
(141) linetypeName = "CENTER"
(142) ' Load "CENTER" line type from acad.lin file
(143) On Error Resume Next ' trap any load errors
(144) ThisDrawing.Linetypes.Load linetypeName, "acad.lin"
(145) linef.Linetype = linetypeName
(146)End If
(147) Set blockRefObj = ThisDrawing.ModelSpace.InsertBlock(insertPt, mx, 1#, 1#,
1#, 0)
(148) ThisDrawing.Regen acActiveViewport
(149)""""GB197
(150)End If

```

```

(151)ThisDrawing.SetVariable "osmode", sysOSMODE

```

說明：(151) 這條程式用來還原系統變數。

```

(152)End Sub

```

● 程序：UserForm1_Initialize()

作用：初始一交談框，於下拉式列示框中加入銷的標準與參數。

```

(153)Private Sub UserForm_Initialize()
(154)l_up = 0
(155)l_low = 0
(156)a = 0
(157)c = 0
(158)d = 0

```

說明：設定初始參數。

```

(159)ComboBox1.AddItem "GB 199-86", 0
(160)ComboBox1.AddItem "GB 197-86", 1
(161)ComboBox1.ListIndex = 0

```

說明：在下拉式列示框 combobox1 中加入銷的標準型號，並設定預設值為

0。

```
(162)ComboBox2.AddItem "3", 0  
(163)ComboBox2.AddItem "4", 1  
(164)ComboBox2.AddItem "5", 2  
(165)ComboBox2.AddItem "6", 3  
(166)ComboBox2.AddItem "8", 4  
(167)ComboBox2.AddItem "10", 5  
(168)ComboBox2.AddItem "12", 6  
(169)ComboBox2.AddItem "16", 7  
(170)ComboBox2.AddItem "20", 8  
(171)ComboBox2.AddItem "25", 9  
(172)ComboBox2.ListIndex = 0
```

說明：在下拉式列示方塊 combobox2 中加入銷的型號。

```
(173)End Sub
```

● 程序：getdata()

作用：根據傳遞來的參數，從文字檔案中讀取對應的資料。

```
(174)Public Sub getdata(filename, key, p)
```

說明：定義 FSO 物件，檔案以及文字流。其中，參數 filename 代表檔案名稱，key 代表關鍵資料，p 代表關鍵資料所在位置。

```
(175)Dim fso As New FileSystemObject, txtfile, fil1 As file, ts As TextStream
```

```
(176)Set fil1 = fso.GetFile(filename)
```

說明：根據 filename 中的檔案名稱來打開檔案。

```
(177)Set ts = fil1.OpenAsTextStream(ForReading)
```

```
(178)' 打開文字流檔案
```

```
(179)s = ts.ReadLine
```

說明：讀取文字檔案中一行的內容，並將之儲存於 s 中。

```
(180)Do While (1)
(181)    s = ts.ReadLine
(182)    stoa (s)
(183)    If sa(p) = key Then
(184)        Exit Do
(185)    End If
(186)Loop
```

說明：逐行讀取文字檔案，直到在 p 指定位置出現關鍵資料為止。

```
(187)ts.Close
```

說明：關閉文字流檔案。

```
(188)End Sub
```

● 程序：stoa()

作用：以空格為分隔符號，將一串文字轉換為陣列。

```
(189)Public Sub stoa(str As String)
(190)Dim p As Integer, sp As Integer, length As Integer, n As Integer
(191)p = 1
(192)sp = 1
(193)length = 0
(194)n = 0
```

說明：宣告並定義一些變數。

```
(195)    While (p < Len(str) + 2)
(196)        If (Mid(str, p, 1) = " ") Or (Mid(str, p, 1) = "") Then
(197)            sa(n) = CDb1(Mid(str, sp, length))
(198)            n = n + 1
(199)            length = 0
(200)            sp = p + 1
(201)        Else
(202)            length = length + 1
(203)        End If
```

```
(204)      p = p + 1
(205)      Wend
(206)End Sub
```

說明：以空格為分隔符號，將一行文字拆為字串。

● 本範例的執行步驟

1. 載入 pin.dvb 檔案，執行 keys.show 巨集。
2. 依交談式視窗指示與提示文句來輸入所需參數。

15-4 立體 Flange

● 本範例完成圖形

這個範例經用來繪製一個立體的 Flange。本例的特點是會用到三維實體的建立與布林運算。本範例將畫出一個如圖 15-6 所示的立體 Flange：

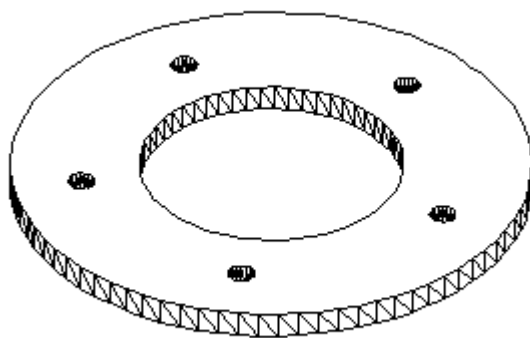


圖 15-6 本程式的立體 Flange 完成圖

本例主要僅用到 flange3d() 主程序。以下部分我們將詳細來講述它們的功能與程式設計的技巧。

- 本範例檔案：Flange3D.dvb（本電子書範例光碟(03M)Samples\Volume4 目錄）
- 主程序：flange3d()（ThisDrawing）

作用：繪製一個 3D 的 Flange。

```
(1)Option Explicit
(2)Public Sub flange3d()
(3)    Dim Outercylinder As Acad3DSolid, InnerCylinder As Acad3DSolid, Hole
(4)As Acad3DSolid
(5)    Dim holes() As Acad3DSolid
(6)    Dim OuterRadius As Double, InnerRadius As Double, HoleRadius As
Double
(7)    Dim center As Variant, center2 As Variant, holecenter As Variant
(8)    Dim height As Double
(9)    Dim HolesNumber As Integer
(10)    Dim HolesPositionRadius As Double
(11)    Dim i As Integer
(12)    Dim util As AcadUtility
```

說明：定義一些包括三維實體的變數。

```
(13)center = ThisDrawing.Utility.GetPoint("請輸入 Flange 的中心點:")
(14)OuterRadius = ThisDrawing.Utility.GetDistance(center, "請輸入 Flange 的外圓
半徑:")
(15)InnerRadius = ThisDrawing.Utility.GetDistance(center, "請輸入 Flange 的內圓
半徑:")
(16)HolesPositionRadius = ThisDrawing.Utility.GetDistance(center, "通過圓孔中心
的中心線位置:")
(17)height = ThisDrawing.Utility.GetDistance("請輸入 Flange 的厚度:")
(18)HolesNumber = ThisDrawing.Utility.GetInteger("圓孔數:")
(19)HoleRadius = ThisDrawing.Utility.GetDistance("圓孔半徑:")
(20)holecenter = ThisDrawing.Utility.PolarPoint(center, 0, HolesPositionRadius)
```

說明：提示操作者輸入 Flange 內徑、外半徑、圓孔所在位置、圓孔半徑、圓孔數與厚度等設計條件參數。

```
(21)Set Outercylinder = ThisDrawing.ModelSpace.AddCylinder(center, OuterRadius,
height)
(22)Set InnerCylinder = ThisDrawing.ModelSpace.AddCylinder(center, InnerRadius,
height)
```


說明：產生兩個圓柱實體。

(23)center2 = center

(24)ReDim holes(HolesNumber)

說明：根據使用者輸入重新定義孔陣列。

(25)For i = 0 To HolesNumber - 1

(26)Set holes(i) = ThisDrawing.ModelSpace.AddCylinder(holecenter, HoleRadius, height)

說明：產生一個孔的圓柱體。

(27)holes(i).Rotate3D center, center2, $2 * 3.14159265 / \text{HolesNumber} * (i + 1)$

說明：旋轉到適當位置。

(28)OuterCylinder.Boolean acSubtraction, holes(i)

說明：在外圓柱中“挖”去孔的圓柱。

(29)Next i

(30)OuterCylinder.Boolean acSubtraction, InnerCylinder

說明：在外圓柱中“挖”去內圓柱。

(31)'Change the viewing direction of the viewport to better see the cylinder

(32)Dim NewDirection(0 To 2) As Double

(33)NewDirection(0) = -1: NewDirection(1) = -1: NewDirection(2) = 1

(34)ThisDrawing.ActiveViewport.Direction = NewDirection

(35)ThisDrawing.ActiveViewport = ThisDrawing.ActiveViewport

(36)ZoomAll

說明：將視埠變換到立體視埠中。

(37)End Sub

● 本範例的執行步驟

1. 載入 flange3d.dvb 檔案，執行 flange3d 巨集。
2. 依交談式視窗指示與提示文句來輸入所需參數。

15-5 與資料庫的結合應用

這一節的範例也是用來繪製螺栓，但與 15-2 節所繪製的螺栓不同。本例的特點在於利用 VBA 存取 ACCESS 資料庫，來讀取螺栓的圖形參數，以繪製出多種類型螺栓。

再與本章 15-3 節的範例相比，從資料庫中讀取資料顯然要比從文字檔案中容易得多，而且無需知道資料的儲存順序。但是將資料儲存在資料庫中，將需要在應用程式中加入相當數量的額外程式。除非您如果不希望有這樣的額外程式，或者資料存取要求不需要用一個與全功能資料庫關聯的額外功能，否則隨著電腦硬體價格的下降及日益增多的資料，用資料庫來管理圖形參數已是一個必然趨勢。

本節範例將教導您使用 ADO 的方法來存取資料庫。Microsoft ActiveX Data Objects (ADO) 將使您能夠撰寫出可透過 OLE DB 來對在資料庫伺服器中的資料進行存取和操作的應用程式。其主要優點是：易於使用、高速、低記憶體支出與佔用磁碟空間較少。ADO 支援適用於使用者端/伺服器端與 Web 端等應用程式的主要功能。ADO 是一個物件模型，它結合了 OLE DB 易於使用的特性以及在諸如 Remote Data Objects(RDO) 和 Data Access Objects(DAO) 的模型中容易找到的通用特性。更重要的是：ADO 包含了所有可以被 OLE DB 標準介面描述的資料類型。換句話說，ADO 物件模型具有可擴展性，它不需要您對自己的元件做任何工作，只要透過一般的 ADO 程式設計介面，您可以視覺化地處理所有的事，即使那些記錄的資訊的格式是您從來沒有見過的。

ADO 在其實際運行中也得到了很高的評價，舉凡記憶體覆蓋，線程安全，分散式事務支援，還可以執行 Web 的遠端資料存取（例如 ADO 同時具有遠端資料服務(RDS) 功能，透過 RDS，您可以在一次往返過程中履行將資料從伺服器移動到使用者端應用程式或 Web 網頁、在使用者端對資料進行處理然後將更新結果傳回伺服器的操作）。作為 Microsoft UDA 策略的一部分，ADO 還試圖成為跨平台資料存取的標準模型。隨著時間的流逝，未來它勢必將取代其他模型（如 DAO）。ADO 也集中了 RDO 和 DAO 所有最好的特性，並將它們重新組織在一個同樣可以提供對事件充分支援，但略微有點不同的物件模型中。

15-5-1 與螺栓資料庫結合的範例

在本例中，我們將為大家示範 ADO 如何作用；與本章第三節類似的，在程式執行之前，要套用 Microsoft ActiveX Data Objects 2.5 Library，否則執行時可能會發生錯誤。

- 本範例完成圖形

本範例將畫出一個如圖 15-7 所示的螺栓：

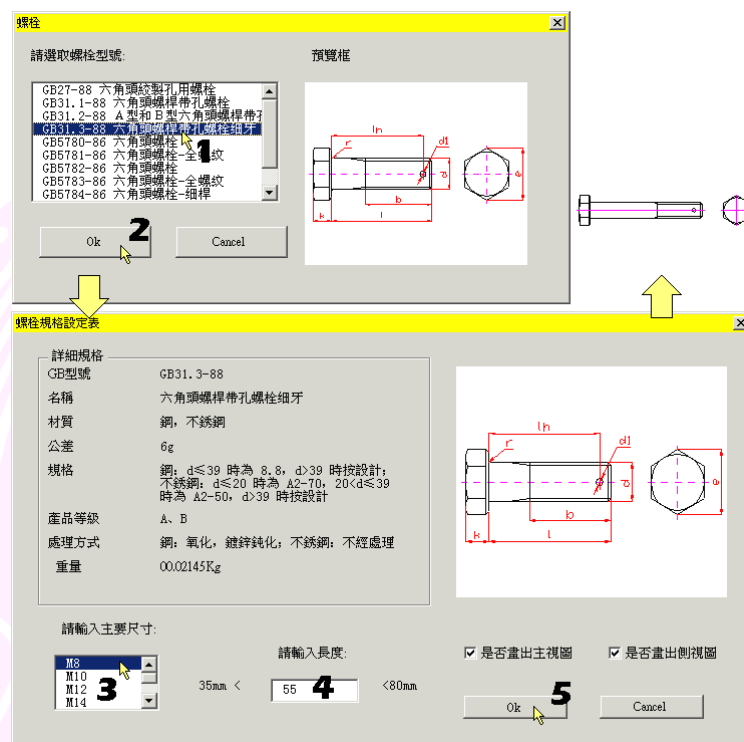


圖 15-7 本程式的螺栓完成圖

本例主要將用到一個程序、四個模組和兩個表單。以下部分我們將詳細來講述它們的功能與程式設計的技巧（本範例的重點技巧說明採註解式）。

- 本範例檔案：Bolt.dvb （本電子書範例光碟(03M)Samples\Volume4 目錄）
Bolt.mdb （本電子書範例光碟(03M)Samples\Volume4 目錄）
*.wmf （本電子書範例光碟(03M)Samples\Volume4 目錄）

- 程序：ThisDrawing

(1)Option Explicit

- 模組：main

作用：呼叫其他模組和表單，是主模組。

```
Public conn As New ADODB.Connection '定義新連接
Public cmd As New ADODB.Command '定義指令
Public rs As New ADODB.Recordset, rs2 As New ADODB.Recordset '定義兩
筆記錄集
Public gb As String '設定 gb 變數來儲存型號
Public exitf1, exitf2 As Boolean '用來判斷是否要退出程式
Public mainview As Boolean, sideview As Boolean '用來判斷是否要主視
圖還是側視圖
Public mainblock As AcadBlock, sideblock As AcadBlock '定義主視圖塊和
側視圖塊
Public mainblockname As String, sideblockname As String '定義主視圖塊
名稱和側視圖塊名稱
Public blength As Double '將螺栓長度存於 blength 變數中
Public mn As String '儲存類似“M？”的螺栓類型
Public Sub boltType() '設定資料類型及檔案路徑，這裡用的是 ACCESS 資料
庫，當然除了 ACCESS，常用的資料庫均可用 ADO 打開，具體的格式請參
考下一節。

'設定螺栓的 Access 資料庫型式與檔案路徑
driver = "{Microsoft Access Driver (*.mdb)}"
dbq = "d:/(03M)Samples/Volume4/bolt.mdb"

'連接至資料庫
conn.Open "driver=" & driver & ";dbq=" & dbq

'設定 SQL 指令
Set cmd.ActiveConnection = conn
cmd.CommandText = "SELECT * from btype" 'operate in table "btype"

'經由已定義的 SQL 來擷取記錄組(列表)
rs.CursorLocation = adUseClient
rs.Open cmd, , adOpenStatic, adLockBatchOptimistic

'呼叫選擇型號表單
UserForm1.Show
```

```
If exitf1 Then '如果點取了「Cancle」鈕就跳出程式
rs.Close      '關閉記錄組以釋放記憶體
conn.Close
Exit Sub
End If
```

'呼叫選擇詳細規格表單

```
UserForm2.Show
```

```
If exitf2 Then '如果點取了「Cancle」鈕就跳出程式
rs.Close      '關閉記錄組以釋放記憶體
conn.Close
Exit Sub
End If
```

'根據使用者的選擇來建立螺栓圖塊

```
Block.DefineBlock
```

```
rs.Close '關閉資料庫的記錄組和連接
conn.Close
ThisDrawing.Regen acActiveViewport '重新產生圖形
End Sub
```

● 模組：block

作用：定義主視圖塊和側視圖塊

```
Public Sub DefineBlock()
```

```
Dim insertPt(0 To 2) As Double
```

```
If mainview = True Then '定義主視圖圖塊名稱
mainblockname = "bbm" & rs(0) & mn
End If
If sideview = True Then '定義側視圖圖塊名稱
sideblockname = "bbs" & rs(0) & mn
End If
```

```

insertPt(0) = 0: insertPt(1) = 0: insertPt(2) = 0  '建立螺栓圖塊
If (mainblockname <> "") Then
Set mainblock = ThisDrawing.Blocks.Add(insertPt, mainblockname)
End If
If (sideblockname <> "") Then
Set sideblock = ThisDrawing.Blocks.Add(insertPt, sideblockname)
End If

Select Case rs(0)  '建立因選擇不同的螺栓而執行不同的螺栓圖塊畫出
Case "GB27"
    CreateBolt.GB27
Case "GB311"
    CreateBolt.GB311
Case "GB312"
    CreateBolt.GB312
Case "GB313"
    CreateBolt.GB313
Case "GB5780"
    CreateBolt.GB5780
Case "GB5781"
    CreateBolt.GB5781
Case "GB5782"
    CreateBolt.GB5782
Case "GB5783"
    CreateBolt.GB5783
Case "GB5784"
    CreateBolt.GB5784
Case "GB5785"
    CreateBolt.GB5785
Case "GB5786"
    CreateBolt.GB5786
End Select

End Sub

```

● 模組：Insert

作用：插入所畫的螺栓視圖圖塊


```
Public Sub Insert()
```

```
'將圖塊插入到模型空間中
```

```
Dim blkRefObj As AcadBlockReference
```

```
Dim insertPt As Variant
```

```
Dim xscale As Double, yscale As Double
```

```
Dim rotangle As Double
```

```
If mainview = True Then
```

```
insertPt = ThisDrawing.Utility.GetPoint(, "主視圖的插入點:")
```

```
'預設 X 與 Y 方向的縮放比例係數與旋轉角度
```

```
xscale = 1: yscale = 1: rotangle = 0
```

```
On Error Resume Next '如果發生錯誤就使用預設值
```

```
xscale = ThisDrawing.Utility.GetReal("X 軸縮放比例係數:")
```

```
yscale = ThisDrawing.Utility.GetReal("Y 軸縮放比例係數:")
```

```
rotangle = ThisDrawing.Utility.GetReal("旋轉角度:")
```

```
Set blkRefObj = ThisDrawing.ModelSpace.InsertBlock(insertPt, mainblockname,  
xscale, yscale, 1#, rotangle)
```

```
End If
```

```
If sideview = True Then
```

```
insertPt = ThisDrawing.Utility.GetPoint(, "側視圖的插入點:")
```

```
xscale = 1: yscale = 1: rotangle = 0
```

```
On Error Resume Next
```

```
xscale = ThisDrawing.Utility.GetReal("X 軸縮放比例係數:")
```

```
yscale = ThisDrawing.Utility.GetReal("Y 軸縮放比例係數:")
```

```
rotangle = ThisDrawing.Utility.GetReal("旋轉角度:")
```

```
Set blkRefObj = ThisDrawing.ModelSpace.InsertBlock(insertPt, sideblockname,  
xscale, yscale, 1#, rotangle)
```

```
End If
```

```
End Sub
```

● 模組：createbolt

作用：這是模組是最長但也是最規律的程式，因為它包含了所有標準的螺栓的畫出程式（於此，我們僅列出三個）。

```
Public Sub GB27()
```

```
'以下程式將用來畫出螺栓
```

```
Dim util As Object
```

```
Set util = ThisDrawing.Utility '定義一個 utility 來建立點
```

```
Dim mblockf, sblockf As Boolean '定義一個 flag 來判別圖塊是否存在
```

```
mblockf = False
```

```
sblockf = False
```

```
'如果圖塊名稱存在就跳過建立圖塊的程式
```

```
,
```

```
Dim iblock As Integer
```

```
iblock = ThisDrawing.Blocks.Count
```

```
Do While (iblock > 0)
```

```
If ThisDrawing.Blocks.Item(iblock - 1).name = "bbmgb27" & mn Then
```

```
    blockf = True
```

```
End If
```

```
If ThisDrawing.Blocks.Item(iblock - 1).name = "bbsgb27" & mn Then
```

```
    blockf = True
```

```
End If
```

```
iblock = iblock - 1
```

```
Loop
```

```
'定義螺栓的設計參數
```

```
Dim k, r, l3, ds, l2, l, dp, d, e As Double
```

```
'get "l" from global variant "blength"
```

```
l = blength
```

```
'get other parameters from database, some need simple change
```

```
k = rs2("k")
```

```
r = rs2("r")
```

```
l3 = Fix((l - rs2("l_min")) * (rs2("l3_max") - rs2("l3_min")) / (rs2("l_max") -  
rs2("l_min")) + rs2("l3_min"))
```

```
ds = rs2("ds")
l2 = rs2("l2")
dp = rs2("dp")
d = rs2("d")
e = rs2("e")
```

'畫出主視圖的螺栓圖塊

If (mainview = True) And (mblockf = False) Then

'定義一些必要的點變式

```
Dim bp, p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, _
    p14, p15, p16, p17, p18, p19, p20, p21, p22, p23, p24, p25 As Variant
```

'定義弧中心點

```
Dim cenPt1, cenPt2, cenPt3 As Variant
```

'定義畫圖所需的線物件

```
Dim ln1, ln2, ln3, ln4, ln5, ln6, ln7, ln8, ln9, ln10, ln11, ln12, ln13, ln14 As
AcadLine
```

'定義畫圖所需的弧物件

```
Dim a1, a2, a3 As AcadArc
```

'定義一些畫弧所需臨時變式

```
Dim arc1_r, arc1_angs, arc1_ange, arc2_r, arc2_angs, arc2_ange, arc3_r, arc3_angs,
arc3_ange As Double
```

'定義一些畫點所需臨時變式

```
Dim bpx, bpy, p1x, p1y, p2x, p2y, p3x, p3y, p4x, p4y, p5x, p5y, p6x, p6y, p7x, p7y
As Double
```

```
Dim p8x, p8y, p9x, p9y, p10x, p10y, p11x, p11y, p12x, p12y, p13x, p13y, p14x,
p14y, p15x, p15y As Double
```

```
Dim p16x, p16y, p17x, p17y, p18x, p18y, p19x, p19y, p20x, p20y, p21x, p21y, p22x,
p22y, p23x, p24y, p25x, p25y As Double
```

'計算需要的座標點

| | |
|---------------------|-----------------------|
| bpx = 0#: | bpy = 0# |
| p1x = bpx: | p1y = bpy + e * 3 / 8 |
| p2x = bpx: | p2y = bpy |
| p3x = p1x + k / 10: | p3y = bpy + e / 2 |
| p4x = p3x: | p4y = bpy + e / 4 |
| p5x = p3x: | p5y = bpy - e / 4 |
| p6x = bpx + k: | p6y = p3y |

| | |
|-------------------------------------|---|
| p7x = p6x: | p7y = bpy + ds / 2 + r |
| p8x = p6x: | p8y = p4y |
| p9x = p6x: | p9y = bpy |
| p10x = bpx + k + r: | p10y = p7y |
| p11x = p10x: | p11y = bpy + ds / 2 |
| p12x = bpx + k + l3 - (d - dp) / 5: | p12y = p11y |
| p13x = p12x: | p13y = bpy + dp / 2 + (d - dp) * 3 / 10 |
| p14x = p12x: | p14y = bpy |
| p15x = bpx + k + l3: | p15y = bpy + d / 2 |
| p16x = p15x: | p16y = p13y |
| p17x = p15x: | p17y = p14y |
| p18x = bpx + k + l - l2: | p18y = p15y |
| p19x = p18x: | p19y = p16y |
| p20x = p18x: | p20y = bpy + dp / 2 |
| p21x = p18x: | p21y = bpy |
| p22x = bpx + k + l: | p22y = p20y |
| p23x = p22x: | p23y = p21y |
| p24x = bpx - k / 5: | p24y = bpy |
| p25x = bpx + k + l + k / 5: | p25y = bpy |

'建立點

```

util.CreateTypedArray p1, vbDouble, p1x, p1y, 0
util.CreateTypedArray p2, vbDouble, p2x, p2y, 0
util.CreateTypedArray p3, vbDouble, p3x, p3y, 0
util.CreateTypedArray p4, vbDouble, p4x, p4y, 0
util.CreateTypedArray p5, vbDouble, p5x, p5y, 0
util.CreateTypedArray p6, vbDouble, p6x, p6y, 0
util.CreateTypedArray p7, vbDouble, p7x, p7y, 0
util.CreateTypedArray p8, vbDouble, p8x, p8y, 0
util.CreateTypedArray p9, vbDouble, p9x, p9y, 0
util.CreateTypedArray p10, vbDouble, p10x, p10y, 0
util.CreateTypedArray p11, vbDouble, p11x, p11y, 0
util.CreateTypedArray p12, vbDouble, p12x, p12y, 0
util.CreateTypedArray p13, vbDouble, p13x, p13y, 0
util.CreateTypedArray p14, vbDouble, p14x, p14y, 0
util.CreateTypedArray p15, vbDouble, p15x, p15y, 0
util.CreateTypedArray p16, vbDouble, p16x, p16y, 0
util.CreateTypedArray p17, vbDouble, p17x, p17y, 0

```

```
util.CreateTypedArray p18, vbDouble, p18x, p18y, 0
util.CreateTypedArray p19, vbDouble, p19x, p19y, 0
util.CreateTypedArray p20, vbDouble, p20x, p20y, 0
util.CreateTypedArray p21, vbDouble, p21x, p21y, 0
util.CreateTypedArray p22, vbDouble, p22x, p22y, 0
util.CreateTypedArray p23, vbDouble, p23x, p23y, 0
util.CreateTypedArray p24, vbDouble, p24x, p24y, 0
util.CreateTypedArray p25, vbDouble, p25x, p25y, 0
```

'畫出螺栓中的線段

```
Set ln1 = mainblock.AddLine(p1, p2)
Set ln2 = mainblock.AddLine(p3, p6)
Set ln3 = mainblock.AddLine(p4, p8)
Set ln4 = mainblock.AddLine(p6, p9)
Set ln5 = mainblock.AddLine(p11, p12)
Set ln6 = mainblock.AddLine(p12, p14)
Set ln7 = mainblock.AddLine(p12, p15)
Set ln8 = mainblock.AddLine(p15, p17)
Set ln9 = mainblock.AddLine(p15, p18)
Set ln10 = mainblock.AddLine(p13, p19)
Set ln11 = mainblock.AddLine(p18, p21)
Set ln12 = mainblock.AddLine(p20, p22)
Set ln13 = mainblock.AddLine(p22, p23)
```

'畫出螺栓中的弧段

```
cenPt1 = centerPt(p3, p1, p4) '擷取三點中心
arc1_r = distance(cenPt1, p1) '計算弧半徑
arc1_angs = ThisDrawing.Utility.AngleFromXAxis(cenPt1, p3) '計算弧的起始角
arc1_ange = ThisDrawing.Utility.AngleFromXAxis(cenPt1, p4) '計算弧的終止角
Set a1 = mainblock.AddArc(cenPt1, arc1_r, arc1_angs, arc1_ange) '畫出弧
```

```
cenPt2 = centerPt(p4, p2, p5)
arc2_r = distance(cenPt2, p2)
arc2_angs = ThisDrawing.Utility.AngleFromXAxis(cenPt2, p4)
arc2_ange = ThisDrawing.Utility.AngleFromXAxis(cenPt2, p2)
Set a2 = mainblock.AddArc(cenPt2, arc2_r, arc2_angs, arc2_ange)
```

```
cenPt3 = p10
```

```

arc3_r = distance(cenPt3, p7)
arc3_angs = ThisDrawing.Utility.AngleFromXAxis(cenPt3, p7)
arc3_ange = ThisDrawing.Utility.AngleFromXAxis(cenPt3, p11)
Set a3 = mainblock.AddArc(cenPt3, arc3_r, arc3_angs, arc3_ange)

```

'鏡像半個螺栓

```

Dim acadent As AcadEntity '定義一個圖素
For Each acadent In mainblock '逐一鏡像螺栓的每一個圖素
    acadent.Mirror p2, p9
Next acadent

```

```

Set ln14 = mainblock.AddLine(p24, p25) '畫出一條中心線
ln14.Color = acMagenta '改變中心線的顏色
End If

```

'畫出側視圖的螺栓圖塊

```

If (sideview = True) And (sblockf = False) Then

```

```

    Dim center, pt1, pt2, pt3, pt4 As Variant '定義需要的點
    Dim pt1x, pt1y, pt2x, pt2y, pt3x, pt3y, pt4x, pt4y As Double '定義一些變式來
    建立點

```

```

    Dim line1, line2 As AcadLine '定義二條線
    Dim circ As AcadCircle '定義一圓

```

'計算出所需的座標點

```

s = 3 ^ 0.5 / 2 * e
pt1x = s / 2:      pt1y = -e / 4
pt2x = pt1x:      pt2y = e / 4
pt3x = s / 2 + s / 10:  pt3y = 0
pt4x = -pt3x:      pt4y = 0

```

'建立點

```

util.CreateTypedArray center, vbDouble, 0, 0, 0
util.CreateTypedArray pt1, vbDouble, pt1x, pt1y, 0
util.CreateTypedArray pt2, vbDouble, pt2x, pt2y, 0
util.CreateTypedArray pt3, vbDouble, pt3x, pt3y, 0
util.CreateTypedArray pt4, vbDouble, pt4x, pt4y, 0

```



```

'畫出一些線，並設定其中之一的顏色為 magenta
Set line1 = sideblock.AddLine(pt1, pt2)
Set line2 = sideblock.AddLine(pt3, pt4)
line2.Color = acMagenta

'以陣列方式畫出二條線
line1.ArrayPolar 7, 3.14 * 2, center
line2.ArrayPolar 2, 3.14 / 2, center
'畫出圓
Set circ = sideblock.AddCircle(center, s / 2)
End If

'產生圖形
ThisDrawing.Regen

Insertbolt.Insert '插入螺栓圖塊

End Sub
'-----
Public Sub GB311()
    Dim util As Object
    Set util = ThisDrawing.Utility
    Dim mblockf, sblockf As Boolean
    mblockf = False
    sblockf = False

    Dim iblock As Integer
    iblock = ThisDrawing.Blocks.Count
    Do While (iblock > 0)
        If ThisDrawing.Blocks.Item(iblock - 1).name = "bbmgb311" & mn Then
            blockf = True
        End If
        If ThisDrawing.Blocks.Item(iblock - 1).name = "bbsgb311" & mn Then
            blockf = True
        End If
        iblock = iblock - 1
    Loop

```

```

Dim k, r, lh, d1, l, d, e, b As Double
l = blength
k = rs2("k")
r = rs2("r")
lh = Fix((l - rs2("l_min")) * (rs2("lh_max") - rs2("lh_min")) / (rs2("l_max") -
rs2("l_min")) + rs2("lh_min"))
d1 = rs2("d1")
b = rs2("b")
d = rs2("d")
e = rs2("e")

If (mainview = True) And (mblockf = False) Then
Dim bp, p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, _
    p14, p15, p16, p17, p18, p19, p20, p21, p22, p23, p24, p25 As Variant
Dim cenPt1, cenPt2, cenPt3 As Variant
Dim ln1, ln2, ln3, ln4, ln5, ln6, ln7, ln8, ln9, ln10, ln11, ln12, ln13, ln14 As
AcadLine
Dim a1, a2, a3 As AcadArc
Dim c1 As AcadCircle
Dim arc1_r, arc1_angs, arc1_ange, arc2_r, arc2_angs, arc2_ange, arc3_r, arc3_angs,
arc3_ange As Double
Dim bpx, bpy, p1x, p1y, p2x, p2y, p3x, p3y, p4x, p4y, p5x, p5y, p6x, p6y, p7x, p7y
As Double
Dim p8x, p8y, p9x, p9y, p10x, p10y, p11x, p11y, p12x, p12y, p13x, p13y, p14x,
p14y, p15x, p15y As Double
Dim p16x, p16y, p17x, p17y, p18x, p18y, p19x, p19y, p20x, p20y, p21x, p21y, p22x,
p22y, p23x, p24y, p25x, p25y As Double

    bpx = 0#
    bpy = 0#
    p1x = bpx:
    p1y = bpy + e * 3 / 8
    p2x = bpx:
    p2y = bpy
    p3x = p1x + k / 10:
    p3y = bpy + e / 2
    p4x = p3x:
    p4y = bpy + e / 4
    p5x = p3x:
    p5y = bpy - e / 4
    p6x = bpx + k - k / 10:
    p6y = p3y
    p7x = p6x:
    p7y = p4y
    p8x = p6x:
    p8y = bpy
    p9x = bpx + k:
    p9y = bpy + e / 2 - k / 10

```

| | |
|---------------------------------|-----------------------------|
| p10x = p9x: | p10y = bpy + d / 2 + r |
| p11x = p9x: | p11y = bpy |
| p12x = bpx + k + r: | p12y = p10y |
| p13x = p12x: | p13y = bpy + d / 2 |
| p14x = bpx + k + l - b - d / 5: | p14y = p13y |
| p15x = bpx + k + l - b: | p15y = p13y |
| p16x = p15x: | p16y = bpy + d / 2 - 10 / d |
| p17x = p15x: | p17y = bpy |
| p18x = bpx + k + l - d / 10: | p18y = p15y |
| p19x = p18x: | p19y = p16y |
| p20x = p18x: | p20y = bpy |
| p21x = bpx + k + l: | p21y = p19y |
| p22x = p21x: | p22y = bpy |
| p23x = bpx + k + lh: | p23y = bpy |
| p24x = bpx - k / 5: | p24y = bpy |
| p25x = bpx + k + l + k / 5: | p25y = bpy |

```

util.CreateTypedArray p1, vbDouble, p1x, p1y, 0
util.CreateTypedArray p2, vbDouble, p2x, p2y, 0
util.CreateTypedArray p3, vbDouble, p3x, p3y, 0
util.CreateTypedArray p4, vbDouble, p4x, p4y, 0
util.CreateTypedArray p5, vbDouble, p5x, p5y, 0
util.CreateTypedArray p6, vbDouble, p6x, p6y, 0
util.CreateTypedArray p7, vbDouble, p7x, p7y, 0
util.CreateTypedArray p8, vbDouble, p8x, p8y, 0
util.CreateTypedArray p9, vbDouble, p9x, p9y, 0
util.CreateTypedArray p10, vbDouble, p10x, p10y, 0
util.CreateTypedArray p11, vbDouble, p11x, p11y, 0
util.CreateTypedArray p12, vbDouble, p12x, p12y, 0
util.CreateTypedArray p13, vbDouble, p13x, p13y, 0
util.CreateTypedArray p14, vbDouble, p14x, p14y, 0
util.CreateTypedArray p15, vbDouble, p15x, p15y, 0
util.CreateTypedArray p16, vbDouble, p16x, p16y, 0
util.CreateTypedArray p17, vbDouble, p17x, p17y, 0
util.CreateTypedArray p18, vbDouble, p18x, p18y, 0
util.CreateTypedArray p19, vbDouble, p19x, p19y, 0
util.CreateTypedArray p20, vbDouble, p20x, p20y, 0
util.CreateTypedArray p21, vbDouble, p21x, p21y, 0

```

```
util.CreateTypedArray p22, vbDouble, p22x, p22y, 0
util.CreateTypedArray p23, vbDouble, p23x, p23y, 0
util.CreateTypedArray p24, vbDouble, p24x, p24y, 0
util.CreateTypedArray p25, vbDouble, p25x, p25y, 0
```

```
Set ln1 = mainblock.AddLine(p1, p2)
Set ln2 = mainblock.AddLine(p3, p6)
Set ln3 = mainblock.AddLine(p4, p7)
Set ln4 = mainblock.AddLine(p6, p8)
Set ln5 = mainblock.AddLine(p6, p9)
Set ln6 = mainblock.AddLine(p9, p11)
Set ln7 = mainblock.AddLine(p13, p18)
Set ln8 = mainblock.AddLine(p14, p16)
Set ln9 = mainblock.AddLine(p15, p17)
Set ln10 = mainblock.AddLine(p18, p20)
Set ln11 = mainblock.AddLine(p16, p21)
Set ln12 = mainblock.AddLine(p18, p21)
Set ln13 = mainblock.AddLine(p21, p22)
```

```
cenPt1 = centerPt(p3, p1, p4)
arc1_r = distance(cenPt1, p1)
arc1_angs = ThisDrawing.Utility.AngleFromXAxis(cenPt1, p3)
arc1_ange = ThisDrawing.Utility.AngleFromXAxis(cenPt1, p4)
Set a1 = mainblock.AddArc(cenPt1, arc1_r, arc1_angs, arc1_ange)
```

```
cenPt2 = centerPt(p4, p2, p5)
arc2_r = distance(cenPt2, p2)
arc2_angs = ThisDrawing.Utility.AngleFromXAxis(cenPt2, p4)
arc2_ange = ThisDrawing.Utility.AngleFromXAxis(cenPt2, p2)
Set a2 = mainblock.AddArc(cenPt2, arc2_r, arc2_angs, arc2_ange)
```

```
cenPt3 = p12
arc3_r = distance(cenPt3, p10)
arc3_angs = ThisDrawing.Utility.AngleFromXAxis(cenPt3, p10)
arc3_ange = ThisDrawing.Utility.AngleFromXAxis(cenPt3, p13)
Set a3 = mainblock.AddArc(cenPt3, arc3_r, arc3_angs, arc3_ange)
```

```
Dim acadent As AcadEntity
```

For Each acadent In mainblock

 acadent.Mirror p2, p8

Next acadent

Set c1 = mainblock.AddCircle(p23, d1 / 2)

Set ln14 = mainblock.AddLine(p24, p25)

ln14.Color = acMagenta

End If

If (sideview = True) And (sblockf = False) Then

Dim center, pt1, pt2, pt3, pt4 As Variant

Dim pt1x, pt1y, pt2x, pt2y, pt3x, pt3y, pt4x, pt4y As Double

Dim line1, line2 As AcadLine

Dim circ As AcadCircle

$s = 3^{0.5} / 2 * e$

pt1x = s / 2: pt1y = -e / 4

pt2x = pt1x: pt2y = e / 4

pt3x = s / 2 + s / 10: pt3y = 0

pt4x = -pt3x: pt4y = 0

util.CreateTypedArray center, vbDouble, 0, 0, 0

util.CreateTypedArray pt1, vbDouble, pt1x, pt1y, 0

util.CreateTypedArray pt2, vbDouble, pt2x, pt2y, 0

util.CreateTypedArray pt3, vbDouble, pt3x, pt3y, 0

util.CreateTypedArray pt4, vbDouble, pt4x, pt4y, 0

Set line1 = sideblock.AddLine(pt1, pt2)

Set line2 = sideblock.AddLine(pt3, pt4)

line2.Color = acMagenta

line1.ArrayPolar 7, 3.14 * 2, center

line2.ArrayPolar 2, 3.14 / 2, center

Set circ = sideblock.AddCircle(center, s / 2)

End If

ThisDrawing.Regen acActiveViewport

Insertbolt.Insert

End Sub

'-----

Public Sub GB312()

Dim util As Object

Set util = ThisDrawing.Utility

Dim mblockf, sblockf As Boolean

mblockf = False

sblockf = False

Dim iblock As Integer

iblock = ThisDrawing.Blocks.Count

Do While (iblock > 0)

If ThisDrawing.Blocks.Item(iblock - 1).name = "bbmgb312" & mn Then

blockf = True

End If

If ThisDrawing.Blocks.Item(iblock - 1).name = "bbsgb312" & mn Then

blockf = True

End If

iblock = iblock - 1

Loop

Dim k, r, lh, d1, l, d, e, b As Double

l = blength

k = rs2("k")

r = rs2("r")

lh = Fix((l - rs2("l_min")) * (rs2("lh_max") - rs2("lh_min")) / (rs2("l_max") - rs2("l_min"))) + rs2("lh_min"))

d1 = rs2("d1")

b = rs2("b")

d = rs2("d")

e = rs2("e")

If (mainview = True) And (mblockf = False) Then


```

Dim bp, p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, _
    p14, p15, p16, p17, p18, p19, p20, p21, p22, p23, p24, p25 As Variant
Dim cenPt1, cenPt2, cenPt3 As Variant
Dim ln1, ln2, ln3, ln4, ln5, ln6, ln7, ln8, ln9, ln10, ln11 As AcadLine
Dim a1, a2, a3 As AcadArc
Dim c1 As AcadCircle
Dim arc1_r, arc1_angs, arc1_ange, arc2_r, arc2_angs, arc2_ange, arc3_r, arc3_angs,
arc3_ange As Double
Dim bpx, bpy, p1x, p1y, p2x, p2y, p3x, p3y, p4x, p4y, p5x, p5y, p6x, p6y, p7x, p7y
As Double
Dim p8x, p8y, p9x, p9y, p10x, p10y, p11x, p11y, p12x, p12y, p13x, p13y, p14x,
p14y, p15x, p15y As Double
Dim p16x, p16y, p17x, p17y, p18x, p18y, p19x, p19y, p20x, p20y, p21x, p21y, p22x,
p22y As Double

```

```

bpx = 0#:          bpy = 0#
p1x = bpx:         p1y = bpy + e * 3 / 8
p2x = bpx:         p2y = bpy
p3x = p1x + k / 10: p3y = bpy + e / 2
p4x = p3x:         p4y = bpy + e / 4
p5x = p3x:         p5y = bpy - e / 4
p6x = bpx + k:     p6y = p3y
p7x = p6x:         p7y = bpy + d / 2 - d / 10 + r
p8x = p6x:         p8y = p4y
p9x = p6x:         p9y = bpy
p10x = bpx + k + r: p10y = p7y
p11x = p10x:       p11y = bpy + d / 2 - d / 10
p12x = bpx + k + l - b - d / 10: p12y = p11y
p13x = p12x:       p13y = bpy
p14x = bpx + k + l - b: p14y = bpy + d / 2
p15x = p14x:       p15y = p12y
p16x = p14x:       p16y = bpy
p17x = bpx + k + l: p17y = p14y
p18x = p17x:       p18y = p12y
p19x = p17x:       p19y = bpy
p20x = bpx + k + lh: p20y = bpy

```

$p21x = bpx - k / 5:$ $p21y = bpy$
 $p22x = bpx + k + 1 + k / 5:$ $p22y = bpy$

util.CreateTypedArray p1, vbDouble, p1x, p1y, 0
util.CreateTypedArray p2, vbDouble, p2x, p2y, 0
util.CreateTypedArray p3, vbDouble, p3x, p3y, 0
util.CreateTypedArray p4, vbDouble, p4x, p4y, 0
util.CreateTypedArray p5, vbDouble, p5x, p5y, 0
util.CreateTypedArray p6, vbDouble, p6x, p6y, 0
util.CreateTypedArray p7, vbDouble, p7x, p7y, 0
util.CreateTypedArray p8, vbDouble, p8x, p8y, 0
util.CreateTypedArray p9, vbDouble, p9x, p9y, 0
util.CreateTypedArray p10, vbDouble, p10x, p10y, 0
util.CreateTypedArray p11, vbDouble, p11x, p11y, 0
util.CreateTypedArray p12, vbDouble, p12x, p12y, 0
util.CreateTypedArray p13, vbDouble, p13x, p13y, 0
util.CreateTypedArray p14, vbDouble, p14x, p14y, 0
util.CreateTypedArray p15, vbDouble, p15x, p15y, 0
util.CreateTypedArray p16, vbDouble, p16x, p16y, 0
util.CreateTypedArray p17, vbDouble, p17x, p17y, 0
util.CreateTypedArray p18, vbDouble, p18x, p18y, 0
util.CreateTypedArray p19, vbDouble, p19x, p19y, 0
util.CreateTypedArray p20, vbDouble, p20x, p20y, 0
util.CreateTypedArray p21, vbDouble, p21x, p21y, 0
util.CreateTypedArray p22, vbDouble, p22x, p22y, 0

Set ln1 = mainblock.AddLine(p1, p2)
Set ln2 = mainblock.AddLine(p3, p6)
Set ln3 = mainblock.AddLine(p4, p8)
Set ln4 = mainblock.AddLine(p6, p9)
Set ln5 = mainblock.AddLine(p11, p18)
Set ln6 = mainblock.AddLine(p12, p13)
Set ln7 = mainblock.AddLine(p12, p14)
Set ln8 = mainblock.AddLine(p14, p16)
Set ln9 = mainblock.AddLine(p14, p17)
Set ln10 = mainblock.AddLine(p17, p19)

cenPt1 = centerPt(p3, p1, p4)

```

arc1_r = distance(cenPt1, p1)
arc1_angs = ThisDrawing.Utility.AngleFromXAxis(cenPt1, p3)
arc1_ange = ThisDrawing.Utility.AngleFromXAxis(cenPt1, p4)
Set a1 = mainblock.AddArc(cenPt1, arc1_r, arc1_angs, arc1_ange)

```

```

cenPt2 = centerPt(p4, p2, p5)
arc2_r = distance(cenPt2, p2)
arc2_angs = ThisDrawing.Utility.AngleFromXAxis(cenPt2, p4)
arc2_ange = ThisDrawing.Utility.AngleFromXAxis(cenPt2, p2)
Set a2 = mainblock.AddArc(cenPt2, arc2_r, arc2_angs, arc2_ange)

```

```

cenPt3 = p10
arc3_r = distance(cenPt3, p7)
arc3_angs = ThisDrawing.Utility.AngleFromXAxis(cenPt3, p7)
arc3_ange = ThisDrawing.Utility.AngleFromXAxis(cenPt3, p11)
Set a3 = mainblock.AddArc(cenPt3, arc3_r, arc3_angs, arc3_ange)

```

```

Dim acadent As AcadEntity
For Each acadent In mainblock
    acadent.Mirror p2, p9
Next acadent

```

```

Set ln11 = mainblock.AddLine(p21, p22)
ln11.Color = acMagenta

```

```

Set c1 = mainblock.AddCircle(p20, d1 / 2)
End If

```

```

If (sideview = True) And (sblockf = False) Then
    Dim center, pt1, pt2, pt3, pt4 As Variant
    Dim pt1x, pt1y, pt2x, pt2y, pt3x, pt3y, pt4x, pt4y As Double
    Dim line1, line2 As AcadLine
    Dim circ1, circ2 As AcadCircle

```

```

s = 3 ^ 0.5 / 2 * e
pt1x = s / 2:          pt1y = -e / 4
pt2x = pt1x:          pt2y = e / 4
pt3x = s / 2 + s / 10: pt3y = 0

```

pt4x = -pt3x: pt4y = 0

```
util.CreateTypedArray center, vbDouble, 0, 0, 0
util.CreateTypedArray pt1, vbDouble, pt1x, pt1y, 0
util.CreateTypedArray pt2, vbDouble, pt2x, pt2y, 0
util.CreateTypedArray pt3, vbDouble, pt3x, pt3y, 0
util.CreateTypedArray pt4, vbDouble, pt4x, pt4y, 0
```

```
Set line1 = sideblock.AddLine(pt1, pt2)
Set line2 = sideblock.AddLine(pt3, pt4)
line2.Color = acMagenta
```

```
line1.ArrayPolar 7, 3.14 * 2, center
line2.ArrayPolar 2, 3.14 / 2, center
```

```
Set circ1 = sideblock.AddCircle(center, s / 2)
Set circ2 = sideblock.AddCircle(center, s * 2 / 5)
End If
```

```
ThisDrawing.Regen acActiveViewport
```

```
Insertbolt.Insert
```

```
End Sub
```

'-----

... 以下程式均與上示的程式相同，只是規格不同

● 表單：UserForm1

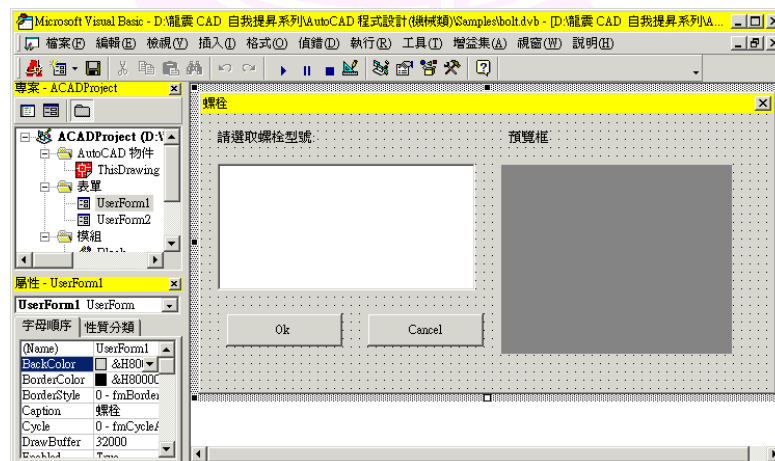


圖 15-8 UserForm1 的外觀圖示

作用：讓使用者選擇螺栓類型

```
Dim li As Integer
```

```
Private Sub CommandButton1_Click()
```

'如果使用者選擇 OK 就執行以下程式段

```
gb = rs(0) '從資料庫中取出 GB 的型號名稱
```

```
Unload Me
```

```
End Sub
```

```
Private Sub CommandButton2_Click()
```

'如果使用者選擇 Cancel 則執行以下程式段

```
exitf1 = True '設定 exitf1 為 true 以退出程式
```

```
Unload Me
```

```
End Sub
```

```
Private Sub ListBox1_Click()
```

'如果使用者用列表框選出螺栓類型，則執行以下程式段

```
li = ListBox1.ListIndex '儲存列表索引編號到 li 變數中
```

'使記錄號指向對應 li 的索引編號

```
rs.MoveFirst
```

```
rs.Move li
```

'擷取影像檔案名稱

```
pfile = " d:\(03M)Samples\Volume4\" & rs(0) & ".wmf"
```

'在預覽框中載入該影像檔內的圖形

```
Image1.Picture = LoadPicture(pfile)
```

End Sub

Private Sub UserForm_Initialize()

'初始表單時執行以下程式段

exitfl = False '將 exitfl 設為 "false"

On Error Resume Next

'以下程式段用來將螺栓類型加入列表框中

Do Until rs.EOF

listext = rs(7) & " " & rs(1)

ListBox1.AddItem listext

rs.MoveNext

Loop

ListBox1.ListIndex = 0 '將預設的列表框索引設為 0

'設定預設的影像檔案是 "gb27.wmf"

Image1.Picture = LoadPicture("d:/(03M)Samples/Volume4/gb27.wmf")

End Sub

● 表單：UserForm2

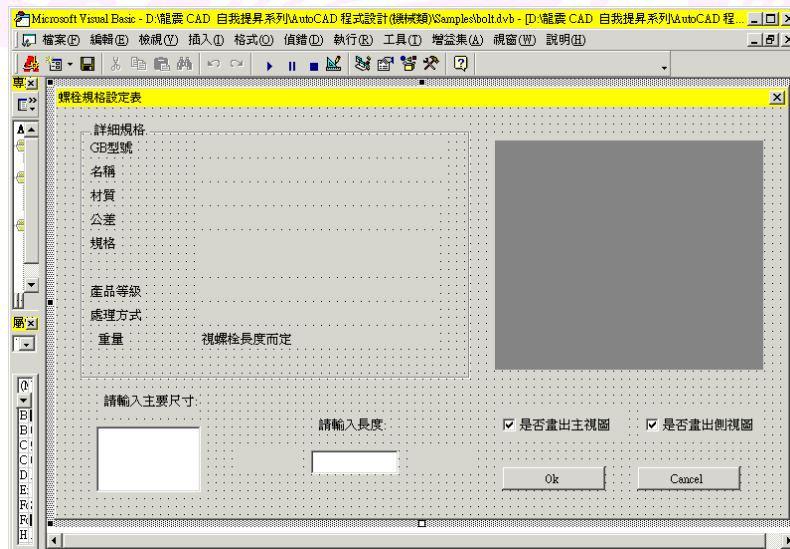


圖 15-9 UserForm1 的外觀圖示

作用：提示使用者輸入螺栓的型號，長度及選擇所需視圖

Dim lmin As Double, lmax As Double '儲存螺栓長度的最大值與最小值

Private Sub CommandButton1_Click()

'如果使用者選擇 OK 就執行以下程式段

'用 mainview 和 sideview 變數來判斷是否主視圖或側視圖已被選擇

mainview = CheckBox1.Value

sideview = CheckBox2.Value

'以下程式將用來先判斷輸入的螺栓長度是否為數字

If Not (IsNumeric(TextBox1.text)) Then

MsgBox "您輸入的並非數字!", vbOKOnly, "無法繼續，因為您必須輸入數字!"

UserForm2.TextBox1.SetFocus

Else

'如果輸入為數字，以下程式再用來判斷數字範圍是否正確

If CDbl(TextBox1.text) < lmin Or CDbl(TextBox1.text) > lmax Then

MsgBox "所輸入的螺栓長並不在允許範圍內!", vbOKOnly, "無法繼續，因為此長度值並不允許!"

UserForm2.TextBox1.SetFocus

Else

Unload Me '如果螺栓長度正確就退去視窗

End If

End If

End Sub

Private Sub CommandButton2_Click()

'如果使用者選擇 Cancel 則執行以下程式段

exitf2 = True '設定 exitf2 為 true 以退出程式

Unload Me

End Sub

'以下程式將根據使用者在列表框中所選擇的型號，確定在輸入長度的輸入

框中顯示長度範圍

```
Private Sub ListBox1_Click()  
li2 = UserForm2.ListBox1.ListIndex '擷取列表的索引  
rs2.MoveFirst '移動記錄組到希望的位置上  
rs2.Move li2  
lmin = rs2("l_min") '讀出資料庫中 "l_min" 欄位的值，並存入變數 lmin 中  
lmax = rs2("l_max") '讀出資料庫中 "l_max" 欄位的值，並存入變數 lmax 中  
l_min.Caption = CStr(lmin) & "mm <" '顯示螺栓長度的最小值  
l_max.Caption = "<" & CStr(lmax) & "mm" '顯示螺栓長度的最大值  
mn = "M" & rs2("d") '以 mn 變數來儲存螺栓的型號  
End Sub
```

```
Private Sub TextBox1_Change()
```

碼
'當輸入框中的文字發生變化時（如刪除或增加一些字時），就執行以下程式

```
'當輸入框中有變化時，則計算對應的重量，並在「重量」標籤中顯示  
If IsNumeric(TextBox1.text) Then  
bweight = CDb(TextBox1.text) / 100 * rs2("weight")  
If bweight >= 1# Then  
weight.Caption = CStr(bweight) & "Kg"  
Else  
weight.Caption = "0" & CStr(bweight) & "Kg"  
End If  
blength = CDb(TextBox1.text) '儲存長度值  
End If  
End Sub
```

'以下程式段用來當初始表單呼叫時

```
Private Sub UserForm_Initialize()
```

'設定表單上的標籤（必須與 Access 資料庫裡的欄位定義一致）

```
gbcode.Caption = rs("GB 型號")  
bname.Caption = rs("名稱")  
material.Caption = rs("材質")  
tolerance.Caption = rs("公差")  
grade.Caption = rs("規格")
```

p_grade.Caption = rs("產品等級")

s_treat.Caption = rs("處理方式")

'在預覽框中載入該影像檔內的圖形

pfile = " d:/(03M)Samples/Volume4/" & rs(0) & ".wmf"

Image2.Picture = LoadPicture(pfile)

'設定 SQL 指令

Set cmd.ActiveConnection = conn

'對列表中代表 rs(0) 的部分進行操作

cmd.CommandText = "SELECT * from " & rs(0)

'執行 SQL 指令來建立記錄組

rs2.CursorLocation = adUseClient

rs2.Open cmd, , adOpenStatic, adLockBatchOptimistic

'增加所有的螺栓型號到列表中

Do Until rs2.EOF

listext = "M" & rs2(0)

ListBox1.AddItem listext

rs2.MoveNext

Loop

ListBox1.ListIndex = 0 '將預設的列表框索引設為 0

TextBox1.SetFocus '讓游標在輸入框中跳動

End Sub

● Access 資料庫部分：BOLT.MDB

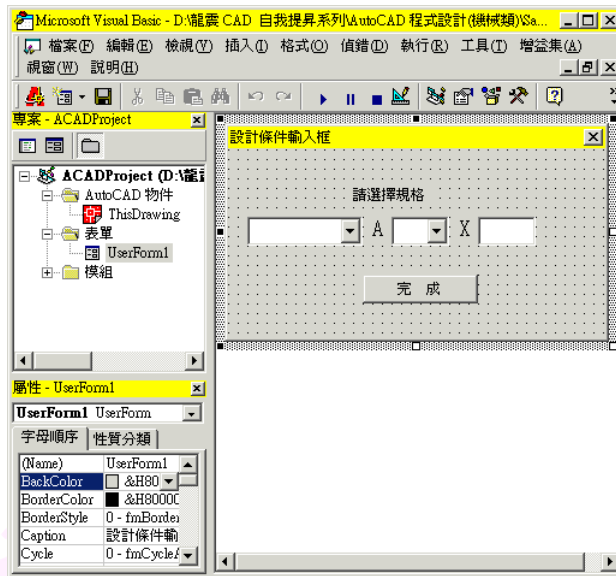


圖 15-10 BOLT.MDB 資料庫的內容

● 本範例的執行步驟

1. 載入 bolt.dvb 檔案，執行 main.bolttype 巨集。
2. 依交談式視窗指示與提示文句來輸入所需參數。

15-5-2 資料庫的設定

通常在使用小型的資料庫時，Access 是一個很好的選擇，它可以全面地支援 SQL（結構化查詢語言）。然而，如果我們需要使用其他資料庫，如：SQL Server 或 dBase 等資料庫，又有什麼東西是需要更改的呢？主要的變動就是有關 ODBC 的設定。

本節將為您補充針對不同資料庫，而需要變動的 ODBC 設定資訊。如下所述：

● Access 部分

```
driver={Microsoft Access Driver (*.mdb)}
dbp=Full Database Path
```

● SQL Server 部分

```
driver={SQL Server}
server=SQL Server Name （這裡的名稱是安裝 SQL Server 時設定的）
```

database=Database Name （安裝於 SQL Server 中的資料庫名稱）
uid=User ID （使用者名稱）
pwd=Password （密碼）

● dBase 部分

driver={Microsoft dBase Driver (*.dbf)}
dbp=Database Path(not include filename)

在使用 dBase 資料庫時，設定 dbq 時並不需包含完整路徑。如：(d:\dbase)，因為這將導致程式將整個 d:\dbase 目錄下的所有資料庫看成一個資料庫，而每個 dbase 則被程式認為是一個列表。

● Excel 部分

driver={Microsoft Excel Driver (*.xls)}
dbq=Full Database path

在 Excel 中，每個工作表均被認為是一個資料庫列表。而對於文字檔案 txt 而言將是：

driver={Microsoft Text Driver (*.txt; *.csv)}
dbq=Database Path(not include filename)

與 dBase 的設定類似，因為程式會將整個路徑下的所有文字檔案看成一個資料庫，而每個文字檔案均被程式認為是一個列表。

由此可以看出：只要設定好與資料庫間的介面，就可以輕易的使用 ADO 來存取資料庫，接著，我們就可以透過 SQL 來對資料庫做查詢、添加與刪除等動作。這確實比其他模型方便多了。有關 ADO 的詳細資訊，如果您已安裝了 Access，就可以在：

C:\Program Files\Common Files\System\ado

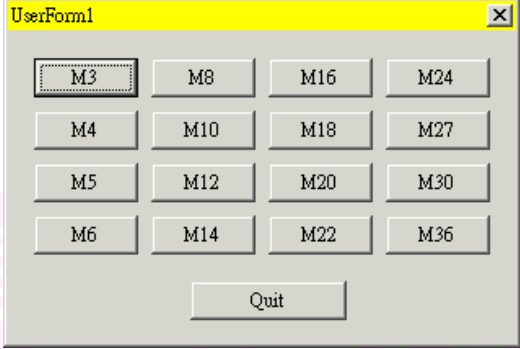
目錄下的 MDACReadme.htm 文件中取得。

啓發性習題

實作題

1. 請改善本章 15-1 節範例的呼叫執行操作介面的親和力。完成圖類似如下：

(解答檔案名稱：AutoSCREW.DVB)



| M3 | M8 | M16 | M24 |
|----|-----|-----|-----|
| M4 | M10 | M18 | M27 |
| M5 | M12 | M20 | M30 |
| M6 | M14 | M22 | M36 |

Quit

2. 請使用 VBA 語法，依 15-5 節範例的模式，自機械法規裡任選一零件來畫出符合法規的零件圖形。