Ⅶ プログラミング

この章では、訓練計画システムのプログラミングの核となる部分について説明する。

1 プログラミング規約

システムを開発するときには、生産性と保守性を向上させるために、プログラミングを行う際の規約を定めておくことが多い。これは一般に、プログラミング規約あるいはコーディング規約と呼ばれている。この規約は、小規模のシステムを１人で開発する場合でも、大規模なシステムを複数人が共同で開発する場合でも、なくてはならない大切なものである。

訓練計画システムの開発においても、当然、プログラミング規約を定めている。その規約は以下のとおりである。

(1) 命名規則

変数、クラス、オブジェクト等の種類を識別するために、変数等には小文字のプレフィクスをつけて、内容を区別する。表Ⅶ－１に命名規則を示す。

表Ⅶ－1　　命名規則

プレフィクス	種　　類
frm	フォーム
btn	コマンドボタン
lbl	ラベル
fra	フレーム
cmb	コンボボックス
scr	スクロールバー
opt	オプションボタン（トグルボタン）
txt	テキストボックス
C	クラス名
m_	クラスのメンバ変数
mf_	クラスのメンバ関数
o_	オブジェクト
l	長整数型変数
i	整数型変数
f	浮動小数点型変数
dt	日付型変数
s	文字列型変数
b	論理型変数

(2) インデント

プログラミングを行うときの段下げ（インデント）は４文字とする。

(3) クラス機能の公開範囲

クラスを作成するときには、外部に公開する情報と、内部でしか利用しない情報を明確に区別するために、外部に公開する変数や関数にはプレフィクスをつけず、内部でしか利用しない変数や関数にはプレフィクスをつける。

イ 外部へ公開する関数

Ｐｕｂｌｉｃ　Ｓｕｂ　［関数名］

→Ｒｅａｄ、Ｗｒｉｔｅｓ・・・

ロ 内部でしか利用しない関数

Ｐｒｉｖａｔｅ　Ｓｕｂ　［関数名］

→ｍｆ＿Ｒｅａｄ、ｍｆ＿Ｗｒｉｔｅｓ・・・

ハ 外部へ公開し、かつ外部へデータを出力する変数

Ｐｕｂｌｉｃ　Ｐｒｏｐｅｒｔｙ　Ｇｅｔ　［変数名］

→Ｄａｔｕｍ・・・

ニ 外部へ公開し、かつ外部からデータを入力する変数

Ｐｕｂｌｉｃ　Ｐｒｏｐｅｒｔｙ　Ｌｅｔ　［変数名］

→Ｄａｔｕｍ・・・

ホ 内部でしか利用しない変数

Ｐｒｉｖａｔｅ　［変数名］

→ｍ＿Ｄａｔｕｍ・・・

2 クラス

訓練計画システムでは、既に述べたように、二つのクラスを作成し、利用している。

(1) ＣＦｉｌｅクラス

ＣＦｉｌｅクラスは、既に説明したように、ファイルの入出力機能をクラスとして実現したものである。システム開発を行う場合、ファイル入出力機能は、非常によく使う機能である。そこで、これをクラスとして提供する。

また、ＣＦｉｌｅクラスの特徴として、利用者がｏｐｅｎ命令やｃｌｏｓｅ命令を発行する必要がないということである。ｏｐｅｎ命令は、データの入出力が最初に行われるときに自動的に発行され、また、ｃｌｏｓｅ命令は、クラスを削除するときに自動的に発行される。

それでは、クラスの例として、ＣＦｉｌｅの詳細な説明を行う。

ｲ　共通部分

機能：属性や、メンバ変数を定義する。

```
BEGIN
  MultiUse = -1                         'True
END

Attribute VB_Name = "CFile"
Attribute VB_Creatable = True
Attribute VB_Exposed = False

Option Explicit
'CFileクラス
'ファイル入出力の機能
'メンバ変数定義
Private m_Name As String                ' ファイル名
Private m_Position As Long              ' ファイル現在ポジジョン
Private m_FileNo As Long                ' ファイル番号
Private m_Datum As String               ' データ
Private m_EndFile As Long               ' EOFフラグ(1:EOF,0:no,2:ファイルなし)
Private m_OpenMode As Long              ' モードフラグ(1:In,0:Out)
```

ﾛ　Ａｄｄｓ関数

機能：Ｎａｍｅで指定した名前のファイルを追加モードでオープンし、Ｄａｔｕｍで指定したデータを１行書き込む。

```
Public Sub Adds()
    mf_Add
End Sub

Private Sub mf_Add()
    Dim l As Long
    If (m_FileNo = -1) Then
        m_OpenMode = 2
        mf_Open
    End If
    Print #m_FileNo, m_Datum
End Sub
```

ﾊ　ＣｈｅｃｋＥｘｉｓｔ関数

機能：Ｎａｍｅで指定した名前のファイルが存在するか否かをチェックする。
ファイルが存在すれば、ＥｎｄＦｉｌｅに０が設定される。
ファイルが存在しなければ、ＥｎｄＦｉｌｅに２が設定される。

```
Public Sub CheckExist()
    mf_CheckExist
End Sub

Private Sub mf_CheckExist()
```

```
    On Error Resume Next
    Dim l As Long
    l = FreeFile
    Open m_Name For Input As l
    If (Err) Then
        Err.Clear
        m_FileNo = -1
        m_EndFile = 2
    Else
        m_FileNo = l
        m_EndFile = 0
    End If
End Sub
```

ニ　Ｒｅａｄ関数

機能：Ｎａｍｅで指定した名前のファイルが存在すれば、そこから１行データを読み込み、それをＤａｔａｍに設定する。
ファイルを最後まで読み込んだときは、ＥｎｄＦｉｌｅに１が設定される。
ファイルが存在しなければ、ＥｎｄＦｉｌｅに２が設定される。

```
Public Sub Read()                          '読み込みをパブリックにする
    mf_Read
End Sub

Private Sub mf_Read()
    Dim l As Long
    If (m_FileNo = -1) Then
        m_OpenMode = 1
        mf_Open
        If (m_FileNo = -2) Then
            m_FileNo = -1
            m_EndFile = 2
            Exit Sub
        End If
    End If
    If (m_EndFile = 1) Then
        Exit Sub
    End If
    On Error Resume Next
    Line Input #m_FileNo, m_Datum
    If (Err) Then
        Err.Clear
        m_EndFile = 1
        Exit Sub
    End If
    If (EOF(m_FileNo)) Then
        m_EndFile = 1
        Exit Sub
    End If
End Sub
```

ホ　Ｔｅｓｔ関数

機能：Ｎａｍｅで指定した名前のファイルを入力モードでオープンし、もし存在しなければ、出力モードでファイルをオープンする。

```
Public Sub Test()
    mf_Test
End Sub

Private Sub mf_Test()
    On Error Resume Next
    Dim l As Long
    l = FreeFile
    Open m_Name For Input As l
    If (Err) Then
        Err.Clear
        l = FreeFile
        Open m_Name For Output As l
    End If
    Close l
End Sub
```

ヘ　Ｗｒｉｔｅｓ関数

機能：Ｎａｍｅで指定した名前のファイルが存在すれば、Ｄａｔｕｍで指定したデータを１行書き込む。

ファイルが存在しなければ、ＥｎｄＦｉｌｅに２が設定される。

```
Public Sub Writes()
    mf_Write
End Sub

Private Sub mf_Write()
    Dim l As Long
    If (m_FileNo = -1) Then
        m_OpenMode = 0
        mf_Open
    End If
    If (m_FileNo = -2) Then
        m_FileNo = -1
        m_EndFile = 2
        Exit Sub
    End If
    Print #m_FileNo, m_Datum
End Sub
```

ト　Ｄａｔｕｍ変数

機能：書き込むデータを指定する。また、読み出すデータが格納される。

```
Public Property Get Datum() As String
    Datum = m_Datum
```

```
End Property

Public Property Let Datum(s As String)
    m_Datum = s
End Property
```

チ　ＥｎｄＦｉｌｅ変数

機能：ファイルのデータがどのような状態かを報告する。

０：ファイルが存在し、ＥＯＦ（ファイルの終わり）を指していない。

１：ファイルが存在し、ＥＯＦ（ファイルの終わり）に達した。

２：ファイルが存在しない。

```
Public Property Get EndFile() As Long
    EndFile = m_EndFile
End Property
```

リ　Ｎａｍｅ変数

機能：オープンするファイル名を指定する。

```
Public Property Let Name(s As String)
    m_Name = s                          ' 変数m_Nameを公開する
End Property
```

ヌ　クラス初期処理関数

機能：オブジェクトが生成されるときに、必ず動作する内部関数である。

```
Private Sub Class_Initialize()
    m_FileNo = -1
End Sub
```

ル　クラス終了処理関数

機能：オブジェクトが削除されるときに、必ず動作する内部関数である。

```
Private Sub Class_Terminate()
    mf_Close
End Sub
```

ヲ　Ｏｐｅｎ関数（非公開）

機能：ファイルに対する入出力に先だって、ファイルのｏｐｅｎを行う非公開関数である。

```
Private Sub mf_Open()
    On Error Resume Next
```

```
    m_FileNo = FreeFile
    m_EndFile = 0
    Select Case m_OpenMode
    Case 0
        Open m_Name For Output As m_FileNo
    Case 1
        Open m_Name For Input As m_FileNo
    Case 2
        Open m_Name For Append As m_FileNo
    End Select
    If (Err) Then
        MsgBox Err.Description
        Err.Clear
        m_EndFile = 2
        m_FileNo = -2
    End If
End Sub
```

ﾘ　Ｃｌｏｓｅ関数（非公開）

機能：ファイルに対する入出力の最後に、ファイルのｃｌｏｓｅを行う非公開関数である。

```
Private Sub mf_Close()
    If (m_FileNo > -1) Then
        Close m_FileNo
    End If
    m_FileNo = -1
End Sub
```

(2) ＣＷｒｉｔｅクラス

ＣＷｒｉｔｅクラスは、画面とプリンタへの出力機能を、クラスとして実現したものである。システム開発を行う場合、画面とプリンタのどちらかに同じイメージを出力することが多い。そこで、これをクラスとして提供する。

それでは、クラスの例として、ＣＷｒｉｔｅの詳細な説明を行う。

ｲ　共通部分

機能：属性や、メンバ変数、内部関数を定義する。

```
'CWriteクラス
'同一イメージの画面表示・プリンタ出力の機能
'外部対応変数
Private m_StepX As Long              ' X方向枠サイズ
Private m_StepY As Long              ' Y方向枠サイズ
Private m_Kind As Long               '0:フォーム
                                     '1:プリンタ
Private m_Canvas As Object           ' 新しい出力オブジェクト
Private m_FileName As String         ' コマンドファイル名
Private m_Message As String          ' 入力文字列
```

```
Private m_ComParm(2) As String          ' パラメタ
Private m_CurString As Long             ' 文字現在長
'内部変数
Private m_Width As Long                 ' X仮想サイズ
Private m_Height As Long                ' Y仮想サイズ
Private m_HorizontalUnit As Long        ' X分割数
Private m_VerticalUnit As Long          ' Y分割数
Private m_MarginX As Long               ' Xマージン
Private m_MarginY As Long               ' Yマージン
Private m_StepSizeX As Long             ' X単位サイズ
Private m_StepSizeY As Long             ' Y単位サイズ
Private m_SizeX As Long                 ' X総サイズ
Private m_SizeY As Long                 ' Y総サイズ
Private m_FontUnitSizeX As Long         ' フォント計算サイズX
Private m_FontUnitSizeY As Long         ' フォント計算サイズY
Dim sCom() As String                    ' ラインコマンド配列
Private m_PrintOffsetX As Long          ' 拡大Xオフセット
Private m_PrintOffsetY As Long          ' 拡大Yオフセット
Private m_PrintRate As Single           ' 拡大縮尺
Private m_PrintPage As Long             ' 拡大画面表示番号
```

ﾛ Ｃｏｍｍｅｎｔ関数

機能：表示する画面又はプリンタで印刷する紙いっぱいに長方形を１個描き、その中に、Ｍｅｓｓａｇｅで指定された文字列を出力する。

```
Public Sub Comment()
    mf_Comment
End Sub

Private Sub mf_Comment()
    mf_FontPrint 0, 0, m_HorizontalUnit - 1, m_VerticalUnit - 1, m_Message
End Sub
```

ﾊ Ｄｒａｗ関数

機能：Ｋｉｎｄ変数で指定されたキャンバス（画面又はプリンタ）に、ＦｉｌｅＮａｍｅで名前を指定されたＳＰ言語命令群に従って描画する。

```
Public Sub Draw()
mf_Draw
End Sub

Public Sub m_Draw()
    Dim l As Long, m As Long, n As Long
    Dim lOrder As Long, lPos As Long
    Dim lStatus As Long      '-1:Normal 1:Shift
    Dim o_File As CFile
    Dim lComNo As Long
'初期処理
    mf_Initialize
```

```
'メイン処理
    lComNo = 6
    ReDim sCom(lComNo)
'枠内表示
    Set o_File = New CFile      ' 新しいcFileオブジェクトを生成する
    o_File.Name = App.Path + "¥" + m_FileName
    Do While (o_File.EndFile = 0)
        For m = 1 To lComNo
            sCom(m) = ""
        Next
        o_File.Read
        If (o_File.EndFile = 2) Then
            Exit Do
        End If
        m_Message = o_File.Datum
    'Debug.Print o_File.EndFile; m_Message
        '分割
        lPos = 1
        lOrder = 1
        lStatus = -1
        For l = 1 To Len(m_Message)
            If (Mid$(m_Message, l, 1) = ",") Then
                If (lStatus = -1) Then
                    sCom(lOrder) = Mid$(m_Message, lPos, l - 1 - lPos + 1)
                    If ("""" = Mid$(m_Message, lPos, 1)) Then
                        If ("""" = Mid$(m_Message, l - 1, 1)) Then
                            sCom(lOrder) = Mid$(sCom(lOrder), 2, Len(sCom(lOrde
r)) - 2)
                        End If
                    End If
                    lPos = l + 1
                    lOrder = lOrder + 1
                End If
            Else
                If l = Len(m_Message) Then
                    sCom(lOrder) = Mid$(m_Message, lPos, l - lPos + 1)
                    If ("""" = Mid$(m_Message, lPos, 1)) Then
                        If ("""" = Mid$(m_Message, l, 1)) Then
                            sCom(lOrder) = Mid$(sCom(lOrder), 2, Len(sCom(lOrde
r)) - 2)
                        End If
                    End If
                    Exit For
                End If
            End If
            If lOrder > lComNo Then
                Exit Do
            End If
        Next
        'コマンド判定
        Select Case sCom(1)
            Case "INIT"
                mf_Initialize
            Case "UNIT"
```

```
                mf_SetUnit (sCom(2)), (sCom(3))
            Case "RECT"
                mf_Rectangle (sCom(2)), (sCom(3)), (sCom(4)), (sCom(5))
            Case "HLINES"
                mf_HorizontalLines (sCom(2)), (sCom(3)), (sCom(4)), (sCom(5)),
(sCom(6))
            Case "VLINES"
                mf_VerticalLines (sCom(2)), (sCom(3)), (sCom(4)), (sCom(5)), (sC
om(6))
            Case "PRINT"
                mf_Print (sCom(2)), (sCom(3)), (sCom(4)), (sCom(5)), mf_Str(sCom
(6))
            Case "FPRINT"
                mf_FontPrint (sCom(2)), (sCom(3)), (sCom(4)), (sCom(5)), mf_Str
(sCom(6))
            Case "LPRINT"
                mf_LeftPrint (sCom(2)), (sCom(3)), (sCom(4)), (sCom(5)), mf_Str
(sCom(6))
            Case "FLPRINT"
                mf_FontLeftPrint (sCom(2)), (sCom(3)), (sCom(4)), (sCom(5)), mf_
Str(sCom(6))
            Case "LINE"
                mf_Line (sCom(2)), (sCom(3)), (sCom(4)), (sCom(5))
            Case "FILL"
                mf_Fill (sCom(2)), (sCom(3)), (sCom(4)), (sCom(5))
            Case "FONT"
                m_CurString = Len(mf_Str(sCom(6)))
                mf_SetFontSize mf_Str(sCom(6)), (sCom(4)) * (sCom(2)), (sCom(5))
 * (sCom(3))
        End Select
    Loop
'終了処理
    Set o_File = Nothing
    mf_Terminate
End Sub
```

ニ　Ｆｉｌｌ Ａｌｌ関数

機能：表示する画面又はプリンタで印刷する紙を、ＳｔｅｐＸ及びＳｔｅｐＹで分割して枠を描く。そして、その枠の中に、Ｍｅｓｓａｇｅで指定された文字列を出力する。

```
Public Sub FillAll()
    mf_FillAll
End Sub

Private Sub mf_FillAll()
    Dim l As Long, m As Long
'初期処理
    mf_Initialize
'メイン処理
    mf_Rectangle 0, 0, m_HorizontalUnit, m_VerticalUnit
```

```
    mf_HorizontalLines 0, 0, m_HorizontalUnit, m_VerticalUnit, 1
    mf_VerticalLines 0, 0, m_HorizontalUnit, m_VerticalUnit, 1
'文字表示
    mf_SetFontSize m_Message, 1, 1
    mf_FontPrint 0, 0, 1, 1, m_Message
    For l = 0 To m_VerticalUnit - 1
        For m = 0 To m_HorizontalUnit - 1
            mf_Print m, l, m + 1, l + 1, m_Message
        Next
    Next
'終了処理
    mf_Terminate
End Sub
```

ホ　Ｇｅｎｅｒａｔｅ関数

機能：表示する画面又はプリンタで印刷する紙に、週間訓練計画の画面を試験的に描く。

```
Public Sub Generate()        '数値生成プログラムをパブリックにする
    mf_Generate
End Sub

Private Sub mf_Generate()        ' Limitまでの数を生成する
    Dim l As Long, m As Long
    Dim lLeft0 As Long, lLeft1 As Long, lLeft2 As Long
    Dim lLeft3 As Long, lCenter As Long, lRight As Long
    Dim lLine0 As Long, lLine1 As Long
'初期処理
    mf_Initialize
'メイン処理
'
'     0
'     |
' 0--lLine0--lLeft1--lLeft2--lLeft3--lCenter--lRight--m_StepX
'     |
'     lLine1
'     |
'     m_StepY
'
    lLeft0 = 1
    lLeft1 = 2
    lLeft2 = 3
    lLeft3 = 4
    lCenter = 14
    lRight = 20
    lLine0 = 1
    lLine1 = 2
    lLineTop = 0
    m_StepX = 21
    m_StepY = 26
    mf_SetUnit m_StepX, m_StepY
    mf_Rectangle lLeft0, lLine0, lRight, m_StepY
    mf_HorizontalLines lLeft0, lLine1, lRight, m_StepY, 3
```

```
    mf_VerticalLines lLeft0, lLine0, lLeft3, m_StepY, 1
    mf_Line lCenter, lLine0, lCenter, m_StepY
    mf_SetFontSize "山", lLeft1 - lLeft0, lLine1 - lLine0
    m_Message = "週間訓練計画"
    mf_SetFontSize m_Message, m_StepX, 1
    mf_PrintMessage 0, 0, m_StepX, lLine0
    m_Message = "月"
    mf_PrintMessage lLeft0, lLine0, lLeft1, lLine1
    m_Message = "日"
    mf_PrintMessage lLeft1, lLine0, lLeft2, lLine1
    m_Message = "曜"
    mf_PrintMessage lLeft2, lLine0, lLeft3, lLine1
    m_Message = "行    事"
    mf_PrintMessage lLeft3, lLine0, lCenter, lLine1
    m_Message = "備  考"
    mf_PrintMessage lCenter, lLine0, lRight, lLine1
'終了処理
    mf_Terminate
End Sub
```

ハ　Ｉｎｉｔｉａｌｉｚｅ関数

機能：Ｋｉｎｄに従って、画面又はプリンタをキャンバスに設定する。また、ＳｔｅｐＸ及びＳｔｅｐＹの指定に応じて、分割する大きさを実際に計算する。

```
Public Sub Initialize()
    mf_Initialize
End Sub

Private Sub mf_Initialize()
    Select Case m_Kind
        Case TYP_FORM
            m_Canvas.Cls        ' 出力オブジェクトをクリアする
            m_SizeX = lComPrintWidth * lComRate ¥ 100
            m_SizeY = lComPrintHeight * lComRate ¥ 100
        Case TYP_PRINTER
            m_SizeX = lComPrintWidth
            m_SizeY = lComPrintHeight
    End Select
    m_MarginX = m_SizeX * 0.05
    m_Canvas.ScaleLeft = -m_MarginX
    m_Width = m_SizeX - m_MarginX * 2
    m_MarginY = m_SizeY * 0.04
    m_Canvas.ScaleTop = -m_MarginY
    m_Height = m_SizeY - m_MarginY * 2
    mf_SetUnit m_StepX, m_StepY
    mf_PrintPage
    m_CurString = 0
End Sub
```

ト　Ｔｅｒｍｉｎａｔｅ関数

機能：キャンバスがプリンタの場合、プリンタにページの終了を指示する。

```
Public Sub Terminate()
    mf_Terminate
End Sub

Private Sub mf_Terminate()
    If m_Kind = TYP_PRINTER Then
        m_Canvas.EndDoc
    End If
End Sub
```

チ　Ｃａｎｖａｓ変数

機能：データを出力するキャンバス（画面又はプリンタ）を指定する。

```
Public Property Set Canvas(aControl As Object)
    Set m_Canvas = aControl
End Property
```

リ　ＦｉｌｅＮａｍｅ変数

機能：ＳＰ言語命令群を格納するファイル名を指定する。

```
Public Property Let FileName(s As String)
    m_FileName = s
End Property
```

ヌ　Ｋｉｎｄ変数

機能：データを出力するキャンバスが、画面かプリンタかを指定する。

０：画面

１：プリンタ

```
Public Property Let Kind(n As Integer)
    m_Kind = n              ' 変数m_Kindを公開する
End Property
```

ル　Ｍｅｓｓａｇｅ変数

機能：ＦｉｌｌＡｌｌ関数で出力する文字列を指定する。

```
Public Property Let Message(s As String)
    m_Message = s
End Property
```

ヲ　ＰｒｉｎｔＰａｇｅ変数

機能：ＰｒｉｎｔＲａｔｅ変数で２００％拡大を指定したときに、どの部分を画面に表示するかを０～８で指定する。

０：左上の１／４

１：上中央の１／４

：

：

```
Public Property Let Printpage(lPage As Long)
    m_PrintPage = lPage
End Property
```

ワ　ＰｒｉｎｔＲａｔｅ変数

機能：画面に表示する場合の拡大率を指定する。

１：１００％（等倍）

２：２００％（２倍拡大）

```
Public Property Let PrintRate(f As Single)
    m_PrintRate = f
End Property
```

カ　ＳｔｅｐＸ変数

機能：画面又はプリンタを分割するときの横方向の分割数を指定する。

```
Public Property Let StepX(n As Integer)
    m_StepX = n                ' 変数m_Stepを公開する
End Property
```

ヨ　ＳｔｅｐＹ変数

機能：画面又はプリンタを分割するときの縦方向の分割数を指定する。

```
Public Property Let StepY(n As Integer)
    m_StepY = n                ' 変数m_Stepを公開する
End Property
```

タ　非公開関数

以下に示す関数は、非公開ではあるが、ＳＰ言語から直接制御することができる。

(イ) Ｆｉｌｌ関数（非公開）

機能：長方形をパターンで埋める。

```
Private Sub mf_Fill(X1 As Long, Y1 As Long, X2 As Long, Y2 As Long)
    m_Canvas.FillStyle = 5
    m_Canvas.Line (X1 * m_StepSizeX, Y1 * m_StepSizeY)-(X2 * m_StepSizeX, Y2 *
                  m_StepSizeY), , B
    m_Canvas.FillStyle = 1
End Sub
```

(ロ) ＦｏｎｔＬｅｆｔＰｒｉｎｔ関数（非公開）

機能：文字列を長方形の枠内に左詰めで表示する。文字列の大きさは計算される。

```
Private Sub mf_FontLeftPrint(X1 As Long, Y1 As Long, X2 As Long, Y2 As Long, s A
s String)
    m_Message = s
    m_CurString = Len(s)
    mf_SetFontSize s, X2 - X1, Y2 - Y1
    mf_LeftPrintMessage X1, Y1, X2, Y2
End Sub
```

(ハ) ＦｏｎｔＰｒｉｎｔ関数（非公開）

機能：長方形の中央に文字列を表示する。文字列の大きさは計算される。

```
Private Sub mf_FontPrint(X1 As Long, Y1 As Long, X2 As Long, Y2 As Long, s As St
ring)
    m_Message = s
    m_CurString = Len(s)
    mf_SetFontSize s, X2 - X1, Y2 - Y1
    mf_PrintMessage X1, Y1, X2, Y2
End Sub
```

(ニ) ＨｏｒｉｚｏｎｔａｌＬｉｎｅｓ関数（非公開）

機能：水平線を表示する。

```
Private Sub mf_HorizontalLines(X1 As Long, Y1 As Long, X2 As Long, Y2 As Long, l
Step As Long)
    Dim l As Long
    For l = Y1 To Y2 Step lStep
        m_Canvas.Line (X1 * m_StepSizeX, l * m_StepSizeY)-(X2 * m_StepSizeX, l *
 m_StepSizeY)
    Next
End Sub
```

(ホ) ＬｅｆｔＰｒｉｎｔ関数（非公開）

機能：左詰めで文字列を表示する。フォント長は以前のものが流用される。

```
Private Sub mf_LeftPrint(X1 As Long, Y1 As Long, X2 As Long, Y2 As Long, s As St
ring)
    m_Message = s
    mf_LeftPrintMessage X1, Y1, X2, Y2
End Sub
```

(ヘ) ＬｅｆｔＰｒｉｎｔＭｅｓｓａｇｅ関数（非公開）

機能：左詰めで文字列を表示する（フォント長調整あり）。

```
Private Sub mf_LeftPrintMessage(X1 As Long, Y1 As Long, X2 As Long, Y2 As Long)
    m_Canvas.CurrentX = m_StepSizeX * X1 + m_StepSizeX ¥ 10
    m_Canvas.CurrentY = m_StepSizeY * Y1 + (m_StepSizeY * (Y2 - Y1) -
m_Canvas.TextHeight(m_Message)) ¥ 2
    If (m_CurString < Len(m_Message)) Then
        mf_SetFontSize m_Message, X2 - X1, Y2 - Y1
        m_Canvas.Print m_Message
        mf_SetFontSize Left$(m_Message, m_CurString), X2 - X1, Y2 - Y1
    Else
        m_Canvas.Print m_Message
    End If
End Sub
```

(ト) Ｌｉｎｅ関数（非公開）

機能：指定された座標間に直線を引く。

```
Private Sub mf_Line(X1 As Long, Y1 As Long, X2 As Long, Y2 As Long)
    m_Canvas.Line (X1 * m_StepSizeX, Y1 * m_StepSizeY)-(X2 * m_StepSizeX, Y2 * m
_StepSizeY)
End Sub
```

(チ) Ｐｒｉｎｔ関数（非公開）

機能：文字列を表示する。フォント長は以前のものが流用される。

```
Private Sub mf_Print(X1 As Long, Y1 As Long, X2 As Long, Y2 As Long, s As Strin
g)
    m_Message = s
    mf_PrintMessage X1, Y1, X2, Y2
End Sub
```

(リ) ＰｒｉｎｔＭｅｓｓａｇｅ関数（非公開）

機能：文字列を長方形の中央に表示する（フォント長調整あり）。

```
Private Sub mf_PrintMessage(X1 As Long, Y1 As Long, X2 As Long, Y2 As Long)
    m_Canvas.CurrentX = m_StepSizeX * X1 + (m_StepSizeX * (X2 - X1) -
```

```
m_Canvas.TextWidth(m_Message)) ¥ 2
    m_Canvas.CurrentY = m_StepSizeY * Y1 + (m_StepSizeY * (Y2 - Y1) -
m_Canvas.TextHeight(m_Message)) ¥ 2
    If (m_CurString < Len(m_Message)) Then
        mf_SetFontSize m_Message, X2 - X1, Y2 - Y1
        m_Canvas.Print m_Message
        mf_SetFontSize Left$(m_Message, m_CurString), X2 - X1, Y2 - Y1
    Else
        m_Canvas.Print m_Message
    End If
End Sub
```

(ヌ) Ｒｅｃｔａｎｇｌｅ関数（非公開）

機能：長方形を表示する。

```
Private Sub mf_Rectangle(X1 As Long, Y1 As Long, X2 As Long, Y2 As Long)
    m_Canvas.Line (X1 * m_StepSizeX, Y1 * m_StepSizeY)-(X2 * m_StepSizeX, Y2 *
                  m_StepSizeY), , B
End Sub
```

(ル) ＳｅｔＦｏｎｔＳｉｚｅ関数（非公開）

機能：フォントの大きさを長方形の枠に入るように調整する。

```
Private Sub mf_SetFontSize(s As String, X As Long, Y As Long)
    m_Canvas.FontSize = 64
    Do While m_Canvas.TextWidth(s) > m_StepSizeX * X * 0.9
        m_Canvas.FontSize = m_Canvas.FontSize - 1
    Loop
    Do While m_Canvas.TextHeight(s) > m_StepSizeY * Y * 0.8 And m_Canvas.FontSiz
e > 6
        m_Canvas.FontSize = m_Canvas.FontSize - 1
    Loop
End Sub
```

(ヲ) ＳｅｔＵｎｉｔ関数（非公開）

機能：表示する画面に設定する仮想的な長方形の縦横の数を指定する。

```
Private Sub mf_SetUnit(X As Long, Y As Long)
    m_HorizontalUnit = X
    m_StepSizeX = m_Width / m_HorizontalUnit * m_PrintRate
    m_VerticalUnit = Y
    m_StepSizeY = m_Height / m_VerticalUnit * m_PrintRate
    m_Canvas.ScaleLeft = -m_MarginX + m_PrintOffsetX
    m_Canvas.ScaleTop = -m_MarginY + m_PrintOffsetY
End Sub
```

(ﾜ) Ｓｔｒ関数（非公開）

機能：文字列の中に%が存在するかをチェックし、存在すればその機能を呼ぶ。

```
Private Function mf_Str(s As String) As String
    If (0 = InStr(s, "%")) Then
        mf_Str = s
    Else
        mf_Str = m_ComParm(Val(Mid$(s, InStr(s, "%") + 1, 1)))
    End If
End Function
```

(ｶ) ＶｅｒｔｉｃａｌＬｉｎｅｓ関数（非公開）

機能：垂直線を表示する。

```
Private Sub mf_VerticalLines(X1 As Long, Y1 As Long, X2 As Long, Y2 As Long, lSt
ep As Long)
    Dim l As Long
    For l = X1 To X2 Step lStep
        m_Canvas.Line (l * m_StepSizeX, Y1 * m_StepSizeY)-(l * m_StepSizeX, Y2 *
 m_StepSizeY)
    Next
End Sub
```

3 既存のＯＣＸの利用

(1) ＯＣＸの利用

Visual BASICでは、ＯＣＸ（OLE Control）というソフトウェア部品が提供されている。これは、ソフトウェアを開発するときに、フォーム上にＯＣＸを配置するだけで、手軽に他人の作成した機能を利用できるというものである。このため、生産性の向上にきわめて効果が高いとされている。

また、ＯＣＸは、ＯＬＥ２に対応している開発言語であれば、いずれからでも利用可能であり、更に、３２ビット化がはかられており、性能も確保されている。

訓練計画システムでは、行事を追加・更新するときに、その開始・終了日を入力したり、帳票出力を行うための選択をするときに、その開始日・終了日を入力したりする。そこで、このＯＣＸの例として、カレンダー機能を実現したＯＣＸを利用し、それを便利なようにカスタマイズする例を紹介する。

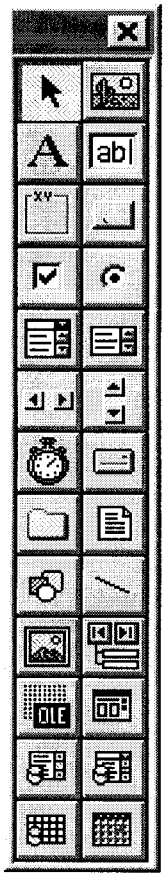
(2) カレンダーＯＣＸ

今回利用するのは、マイクロソフト社が提供しているMicrosoft Access Calendar Control 7.0というＯＣＸである。

ｲ ＯＣＸのVisual BASICへの追加

Microsoft Access Calendar Control 7.0を、Visual BASICの「ツール」メニューの

「カスタムコントロール」で追加すると、図Ⅶ－１のように、通常のツールボックスの右下にカレンダーＯＣＸのボックスが追加される。



図Ⅶ－１　カレンダーＯＣＸ

ㅁ　フォームへのＯＣＸの配置

カレンダーＯＣＸをツールボックスから選択し、フォームに配置する。図Ⅶ－２に、フォームに貼り付けたカレンダーＯＣＸの状態を示す。

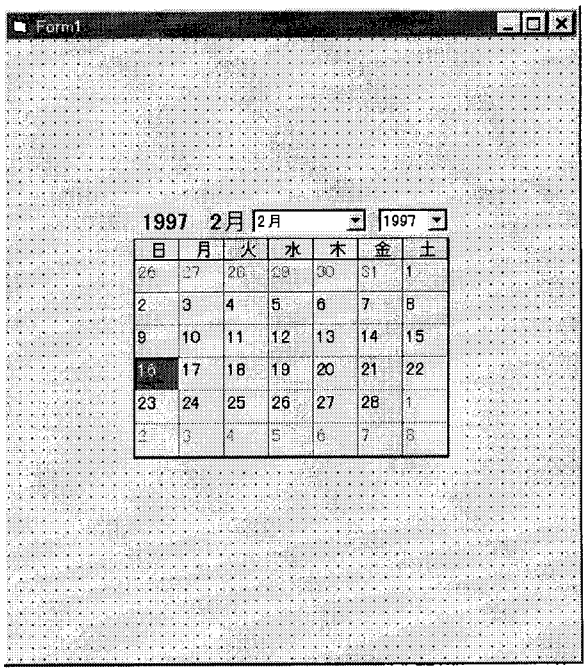
これだけの手順で、プログラムからカレンダー機能を利用する準備ができた。

図Ⅶ－３は、カレンダーＯＣＸの属性を示す、プロパティボックスである。この値をプログラムから変更したり、参照したりすることによって、カレンダーの機能を利用することができる。

(3) カレンダーＯＣＸの欠点

カレンダーＯＣＸは非常によくできているが、訓練計画システムで使用するには、いくつかの欠点がある。

第１は、ＯＣＸが年度対応になっていないということである。訓練計画システムは、年度を区切りとしてデータを管理しているので、利用者が異なる年度のデータを入力できないようにする必要がある。例えば、１９９７年度ということが決まったら、１９９７年４月１日から１９９８年３月３１日のデータのみを扱いたいのである。



図Ⅶ－２　フォームに貼り付けたカレンダーＯＣＸの状態

Calendar1 Calendar	
(About)	
(Custom)	
BackColor	&H00C0C0C0&
DataField	
DataSource	
Day	16
DayFont	(フォント)
DayFontColor	&H00000000&
DayLength	日本語
DragIcon	(なし)
DragMode	0 - Manual
FirstDay	Sun
GridCellEffect	浮き出し
GridFont	(フォント)
GridFontColor	&H00A00000&
GridLinesColor	&H00808080&
Height	2652
HelpContextID	0
Index	
Left	1200
Month	2
MonthLength	日本語
Name	Calendar1
ShowDateSelectors	True
ShowDays	True
ShowHorizontalGrid	True
ShowTitle	True
ShowVerticalGrid	True
TabIndex	0
TabStop	True
Tag	
TitleFont	(フォント)
TitleFontColor	&H00A00000&
Top	1680
Value	97/02/16
ValueIsNull	False
Visible	True
WhatsThisHelpID	0
Width	3372
Year	1997

図Ⅶ－３　カレンダーＯＣＸのプロパティボックス

第２は、月を変更したときに、日の指定が解除されてしまうことである。例えば、４月１０日を指している状態から、月を示すドロップリストで５月へ移動すると、日は指定されなかったことになる。これは大変不便である。

(4) カレンダーＯＣＸと他の部品の組み合わせ

訓練計画システムでカレンダーＯＣＸを利用できるようにするために、年を表示するラベルと、月を表示し、変更することのできるリストボックスを組み合わせることにした。すなわち、利用者は月と日の指定はできるが、年度の指定はできないようにしたのである。これによって、カレンダーＯＣＸが年度対応になっていないという第１番目の欠点を克服した。

また、月が変更された場合、必ず１日を指定するようにした。これによって、第２番目の欠点も克服した。

その具体的な方法は、次のとおりである。

ｲ 部品の決定

以下の四つの部品を組み合わせて年度対応のカレンダーを作成する。

(ｲ) ｆｒａＲａｎｇｅ

カレンダーＯＣＸを乗せるコンテナである。この上に、(ﾛ) ～ (ﾆ) の部品を乗せる。

(ﾛ) ｌｂｌＹｅａｒ

西暦を表示するためのラベルである。訓練計画システムでは、単年度の範囲以外の年度は指定できない。たとえば、１９９７年度の場合、

１９９７年度 → １９９７年4月1日～１９９８年3月３１日

となる。そのため、西暦は、１９９７と１９９８のみ表示できればよいことになる。もともとのカレンダーＯＣＸでは、その指定が自由にできるので、訓練計画システムではわざとラベルにし、ユーザが勝手に変更できないようにする。

(ﾊ) ｃｍｂＭｏｎｔｈ

月を選択し、表示するためのコンボボックスである。

カレンダーＯＣＸでは、

1月
2月
3月
：
１２月

という形のリストが表示され、その中から選択するようになっている。

しかし、年度は４月から始まるので、

４月
５月
６月
：
１２月
１月
２月
３月

と表示された方が都合がよい。

そこで、カレンダーＯＣＸとは別にコンボボックスを用意し、４月から始まるサービスを提供することにする。

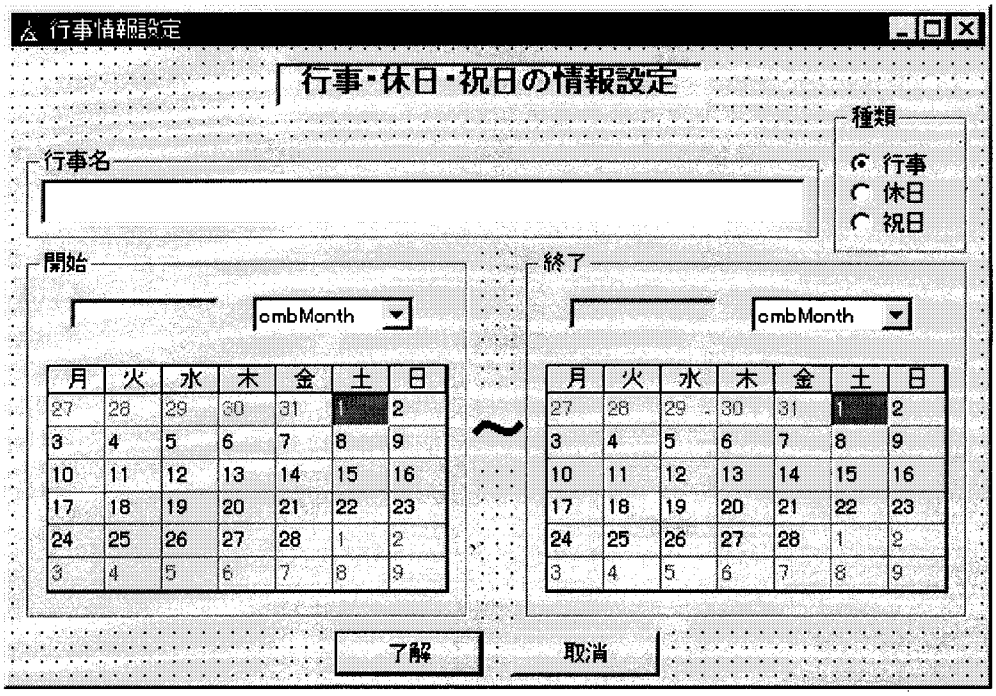
(ﾆ) ｃａｌＥｖｅｎｔ

これは、カレンダーＯＣＸそのものである。

¤ 部品の配置

以下のように、四つの部品を組み合わせて配置する。

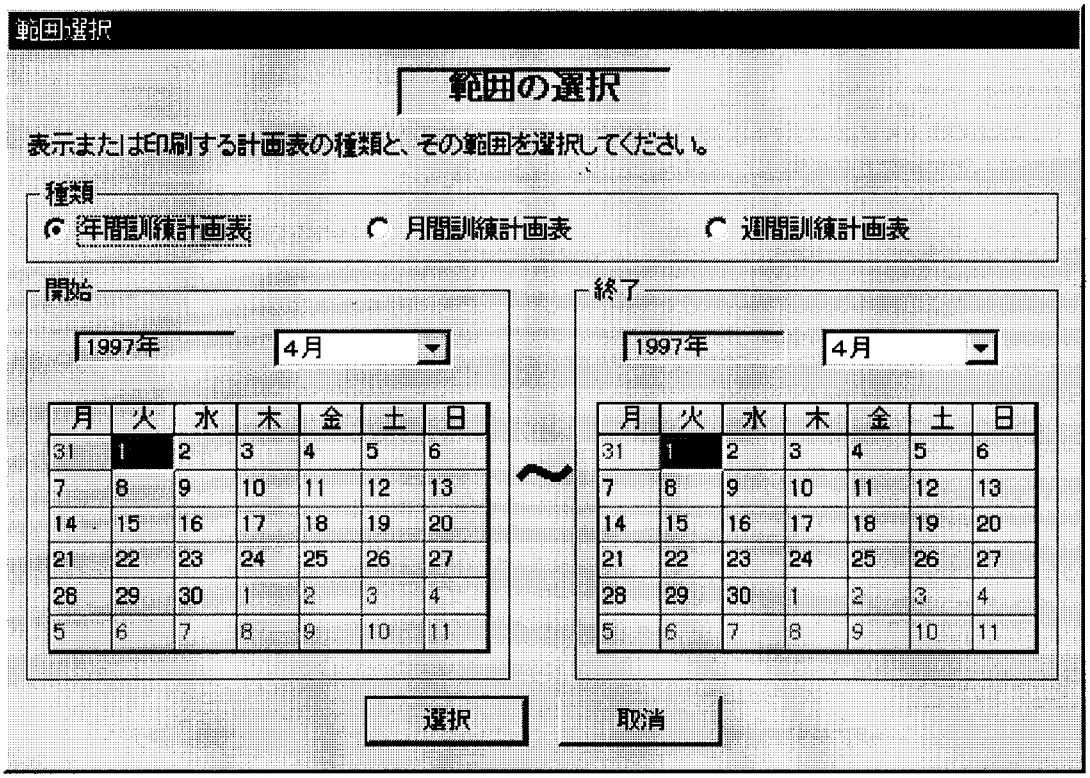
また、開始と終了という二つの年度対応カレンダーにするために、それぞれの部品を２次元の配列とし、０の場合は「開始」、１の場合は「終了」が指定されるようにする。



ハ　実　行

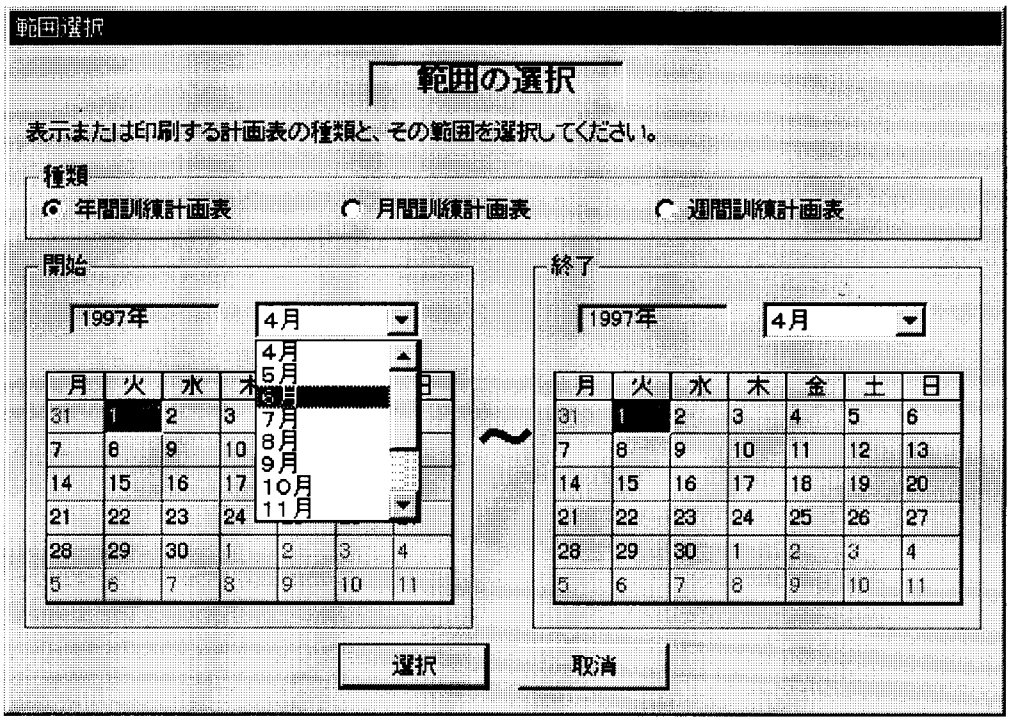
(イ) 実行直後

この年度対応カレンダーを実行すると、以下のように動作する（ただし、年度は１９９７年をあらかじめ指定したものとする。）。



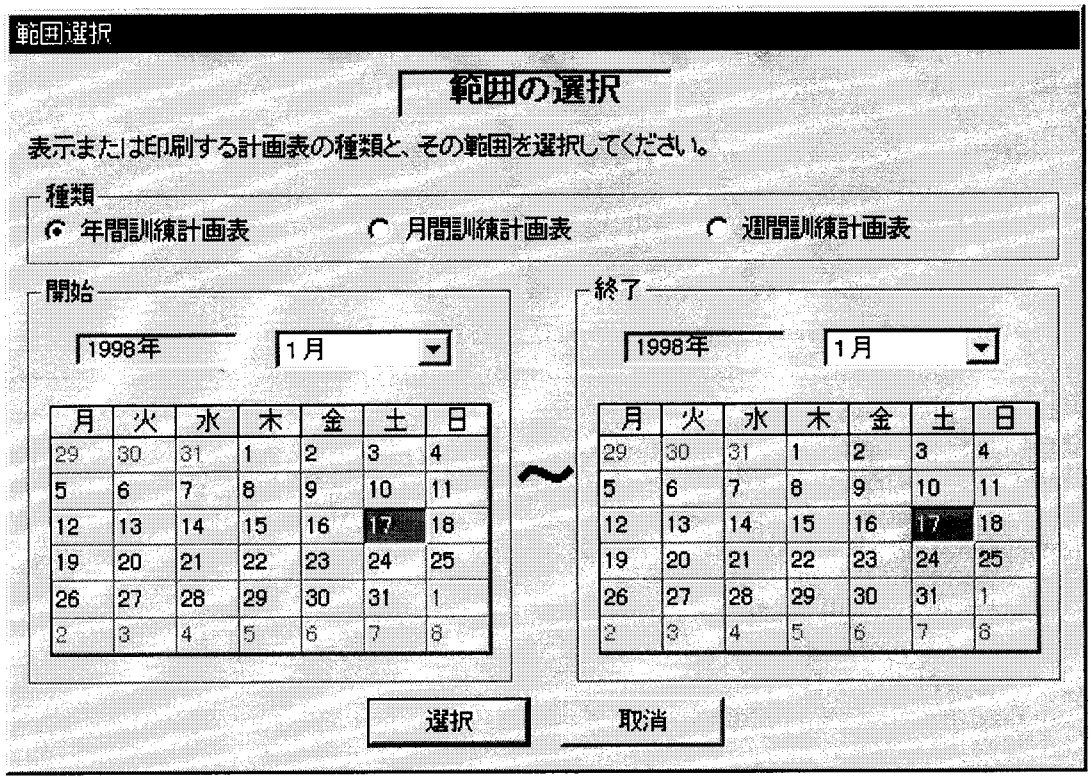
(ロ) 月指定

次に、ドロップリストで６月を選択する。



(ハ) 翌年指定

次に、１９９８年の１月を選択する。



このように、四つの部品が協力しあって、年度指定のカレンダー機能を実現している。

ニ プログラミング

年度指定のカレンダー機能を実現するために追加した機能は、以下のとおりである。

(イ) 共通コード

変数などを定義している部分である。二つのＯＣＸを管理するために、「開始」と「終了」の情報にあわせて２種類の日付用領域を確保している。

```
Option Explicit
Private Type EventData
    EventName As String
    StartDate As Date
    EndDate As Date
    DateType As Long
    EndInput As Boolean
End Type
Private m_Events As EventData
```

(ロ) フォームの初期化

フォームが最初に初期化されるときに動作する部分である。

```
Private Sub Form_Initialize()
```

```
    m_Events.EventName = ""
    m_Events.StartDate = DateSerial(lComNendo, 4, 1)
    m_Events.EndDate = m_Events.StartDate
    m_Events.DateType = 1
Debug.Print "FI"; m_Events.EventName, m_Events.StartDate, m_Events.EndDate, m_Ev
ents.DateType
End Sub
```

(ハ) フォームのロード

４月から１２月と、１月から３月が矛盾なくコンボボックスに表示されるように、コンボボックスのＬｉｓｔＩｎｄｅｘを設定する部分である。

```
Private Sub Form_Load()
    Dim i As Integer
    Move (Screen.Width - Width) / 2, (Screen.Height - Height) / 2
    txtEvent.Text = m_Events.EventName
    SetSelect1 0, lComNendo
    SetSelect1 1, lComNendo
    SetSelect2 0, lComNendo
    SetSelect2 1, lComNendo
    i = m_Events.DateType
    optType(m_Events.DateType).Value = True
    optType_Click i
    calEvent(0).Value = m_Events.StartDate
    If (Month(m_Events.StartDate) >= 4) Then
        cmbMonth(0).ListIndex = Month(m_Events.StartDate) - 4
    Else
        cmbMonth(0).ListIndex = Month(m_Events.StartDate) + 8
    End If
    calEvent_AfterUpdate 0
    calEvent(1).Value = m_Events.EndDate
    If (Month(m_Events.EndDate) >= 4) Then
        cmbMonth(1).ListIndex = Month(m_Events.EndDate) - 4
    Else
        cmbMonth(1).ListIndex = Month(m_Events.EndDate) + 8
    End If
    calEvent_AfterUpdate 1
    Debug.Print "FL"; m_Events.EventName, m_Events.StartDate, m_Events.EndDate,
        m_Events.DateType
End Sub
```

(ニ) 月情報の設定

４月から始まる月をコンボボックスに表示できるようにし、ＩｔｅｍＤａｔａの値と月の数字を一致させている部分である。

```
Private Sub SetSelect1(l As Long, lYear As Long)
    lblYear(l).Caption = Str(lYear) + "年"
    cmbMonth(l).Clear
    cmbMonth(l).AddItem "４月"
```

```
    cmbMonth(1).ItemData(0) = 4
    cmbMonth(1).AddItem "５月"
    cmbMonth(1).ItemData(1) = 5
    cmbMonth(1).AddItem "６月"
    cmbMonth(1).ItemData(2) = 6
    cmbMonth(1).AddItem "７月"
    cmbMonth(1).ItemData(3) = 7
    cmbMonth(1).AddItem "８月"
    cmbMonth(1).ItemData(4) = 8
    cmbMonth(1).AddItem "９月"
    cmbMonth(1).ItemData(5) = 9
    cmbMonth(1).AddItem "１０月"
    cmbMonth(1).ItemData(6) = 10
    cmbMonth(1).AddItem "１１月"
    cmbMonth(1).ItemData(7) = 11
    cmbMonth(1).AddItem "１２月"
    cmbMonth(1).ItemData(8) = 12
    cmbMonth(1).AddItem "１月"
    cmbMonth(1).ItemData(9) = 1
    cmbMonth(1).AddItem "２月"
    cmbMonth(1).ItemData(10) = 2
    cmbMonth(1).AddItem "３月"
    cmbMonth(1).ItemData(11) = 3
End Sub
```

(ホ) その他の情報の設定

フォームをロードした際に、その他の情報を設定する部分である。

```
Private Sub SetSelect2(l As Long, lYear As Long)
    cmbMonth(1).ListIndex = 0
    calEvent(1).Year = lYear
    calEvent(1).Month = 4
    calEvent(1).Day = 1
End Sub
```

(ヘ) カレンダーＯＣＸの日付の変更

もし、カレンダーＯＣＸの日付の部分のうち、前の月の日付が押されたときは、自動的に前の月へ移動しなければならない。

しかし、年度指定のカレンダーでは、前の年度へ移動することは禁止しなければならない。

同様に、カレンダーＯＣＸの日付の部分のうち、次の月の日付が押されたときは、自動的に次の月へ移動しなければならない。

しかし、年度指定のカレンダーでは、次の年度へ移動することも禁止しなければならない。

そこで、年度の範囲以外の部分への移動のときは、年度の範囲内で、最も近い日付けに移動するだけにする。

例えば、１９９７年度において、１９９７年３月２７日が指定されたら、１９９７年４月１日に移動させる。

また、１９９７年度において、１９９７年４月５日が指定されたら、１９９７年３月３１日に移動させる。

```
Private Sub calEvent_AfterUpdate(Index As Integer)
    If (calEvent(Index).Month <> cmbMonth(Index).ItemData(cmbMonth(Index).ListIn
dex)) Then
        If (calEvent(Index).Day < 15) Then
            If (cmbMonth(Index).ListIndex < 11) Then
                cmbMonth(Index).ListIndex = cmbMonth(Index).ListIndex + 1
            Else
                calEvent(Index).Month = 3
                calEvent(Index).Day = 31
            End If
        Else
            If (cmbMonth(Index).ListIndex > 0) Then
                cmbMonth(Index).ListIndex = cmbMonth(Index).ListIndex - 1
            Else
                calEvent(Index).Month = 4
                calEvent(Index).Day = 1
            End If
        End If
    End If
    If (Index = 0) Then
        calEvent(1).Value = calEvent(0).Value
        cmbMonth(1).ListIndex = cmbMonth(0).ListIndex
    Else
        If (calEvent(0).Value > calEvent(1).Value) Then
            calEvent(0).Value = calEvent(1).Value
            cmbMonth(0).ListIndex = cmbMonth(1).ListIndex
        End If
    End If
End Sub
```

(ト) 月コンボボックスの値の変更

月を変更するコンボボックスにおいて、４より小さい月が指定されたときは、西暦の表示を、年度＋１に変更しなければならない。

以下に示すのは、西暦表示を管理する部分のリストである。

```
Private Sub cmbMonth_Click(Index As Integer)
    Dim l As Long
    l = calEvent(Index).Day
    If (cmbMonth(Index).ItemData(cmbMonth(Index).ListIndex) < 4) Then
        lblYear(Index).Caption = Str(lComNendo + 1) + "年"
        calEvent((Index)).Year = lComNendo + 1
    Else
        lblYear(Index).Caption = Str(lComNendo) + "年"
        calEvent((Index)).Year = lComNendo
    End If
```

```
    calEvent((Index)).Month = cmbMonth(Index).ItemData(cmbMonth(Index).ListInde
x)
    calEvent((Index)).Day = 1
    calEvent_AfterUpdate (Index)
End Sub
```

4 独自のＯＣＸの開発

(1) ＯＣＸの開発ツール

ＯＣＸの部品は、Visual Basic等で簡単に利用することができ、きわめて有用である。しかし、どうしても自分の要求に合ったＯＣＸが見つからない場合には、それを開発することも必要になる。開発は、Visual C++で行うのが最も効率的である。

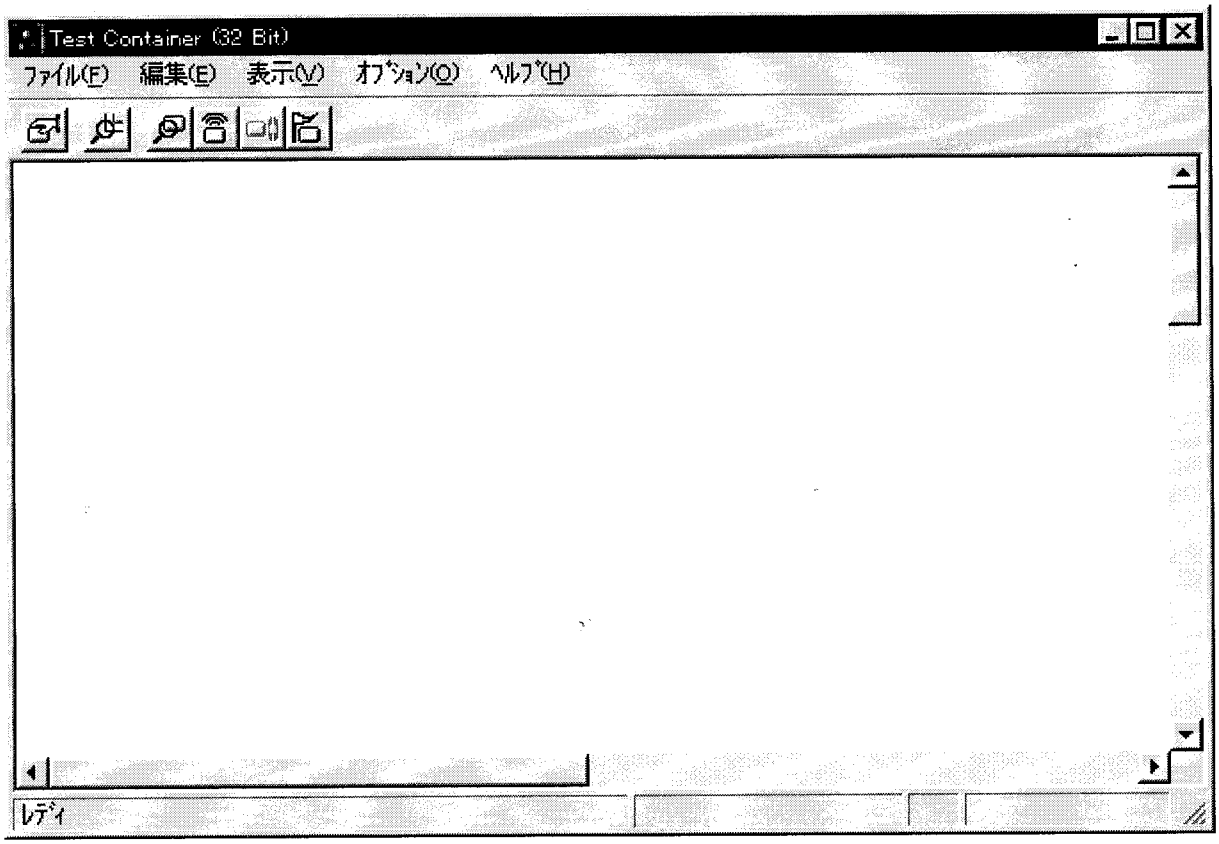
ここでは、独自のＯＣＸ開発の例として、７セギュメントＬＥＤの機能を実現したＯＣＸを開発した例を紹介する。

(2) ７セギュメントＬＥＤ用ＯＣＸ

これは、テストコンテナ機能を利用して表示させている。

ｲ テストコンテナの起動

まず、テストコンテナを起動する。テストコンテナとは、独自に開発したＯＣＸの動作を確認するための、Visual C++のツールである。

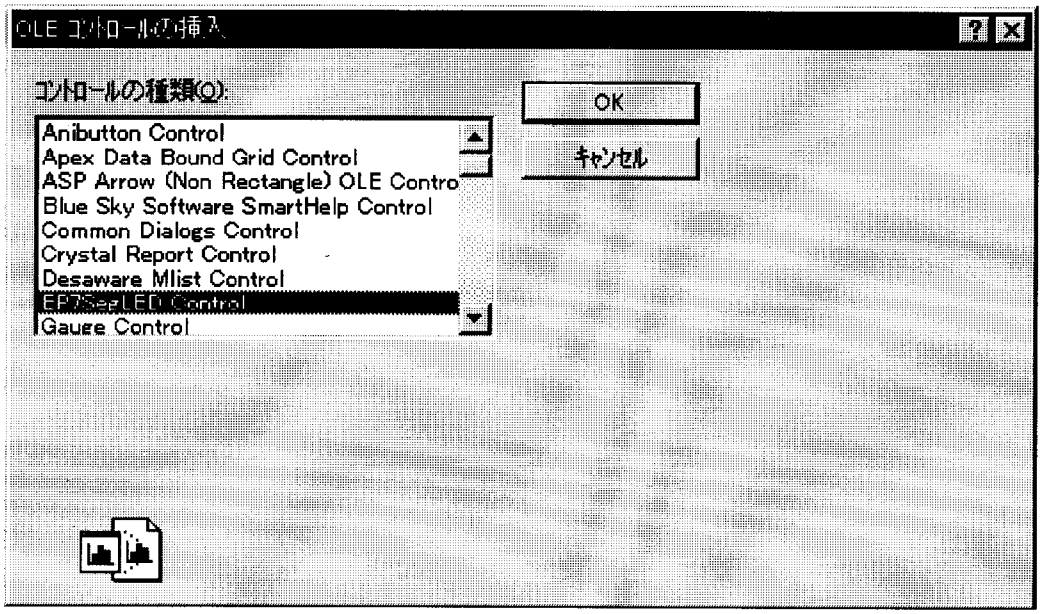


ﾛ　ＯＣＸのセット

ツールバーの左端のボタンを押して、ＯＣＸをセットする。

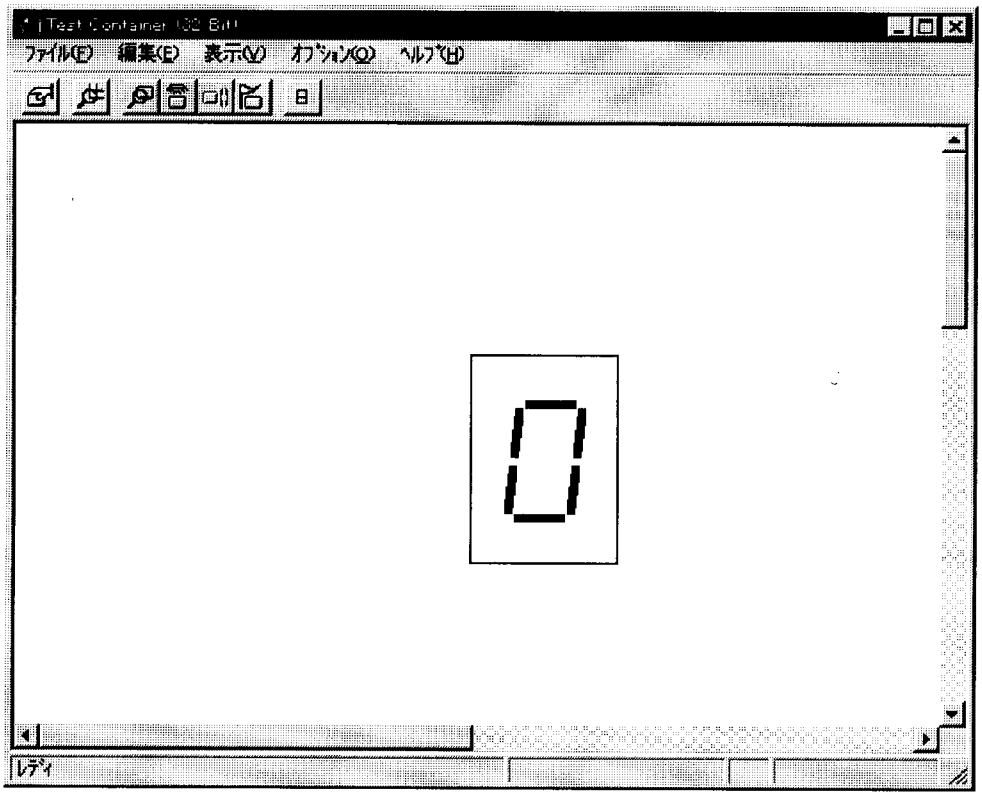
ﾊ　ＯＣＸの選択

ＥＰ７ＳｅｇＬＥＤ　Ｃｏｎｔｒｏｌを選択する。



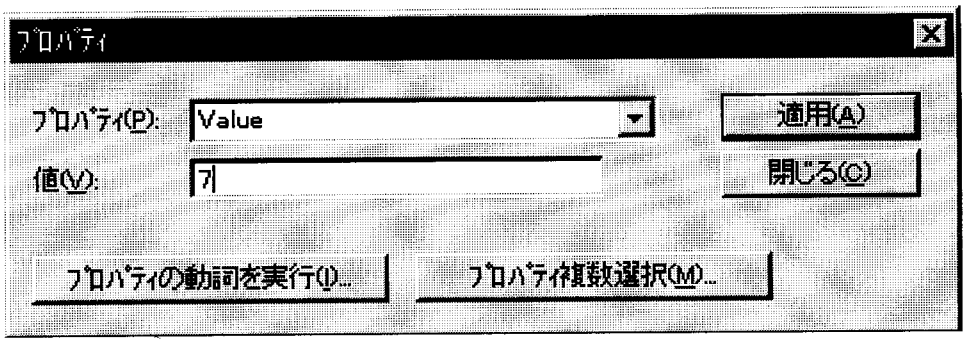
ﾆ　ＯＣＸの初期状態

ＯＣＸの初期の状態がセットされる。７セギュメントＬＥＤの初期値である０が表示されている。



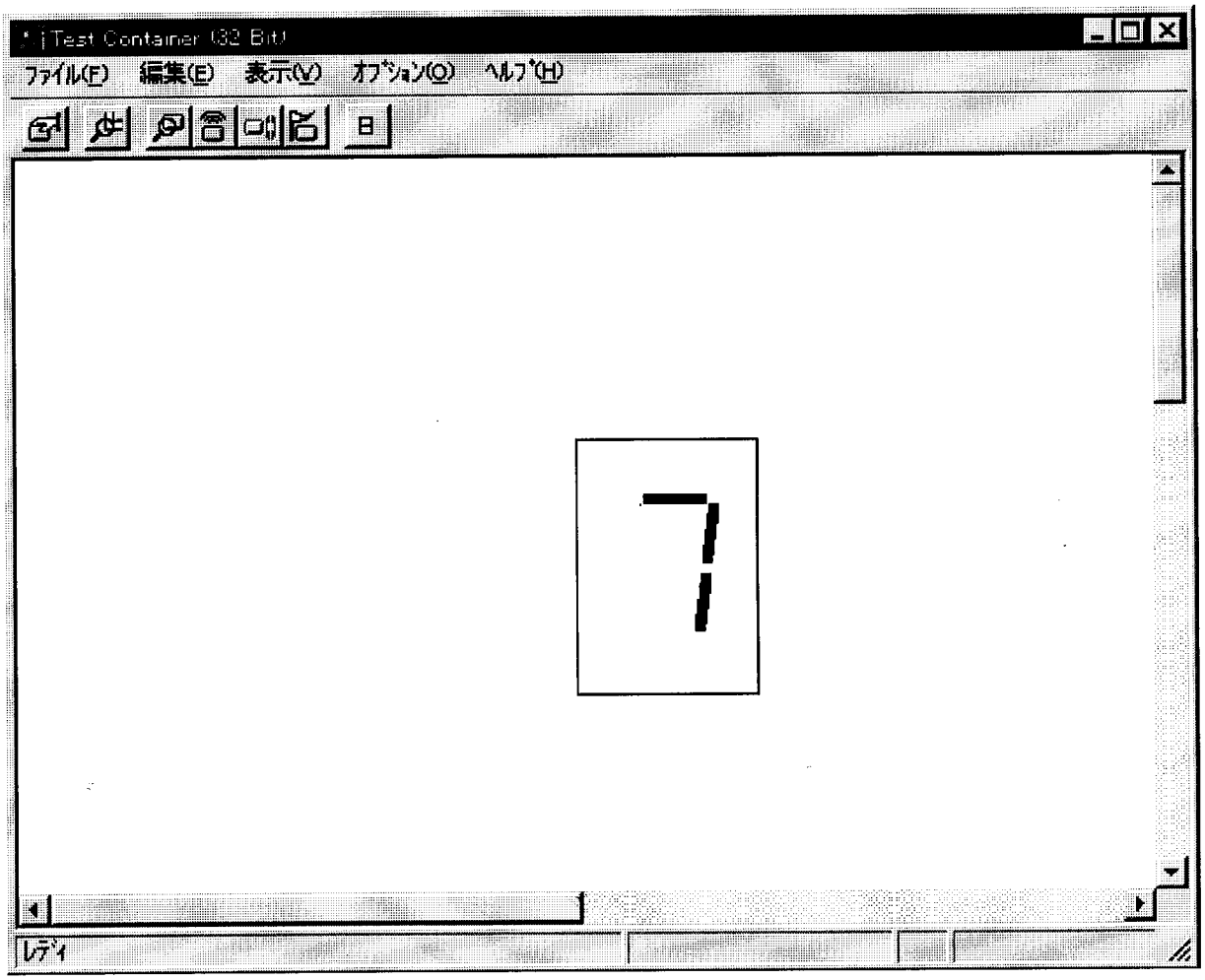
ホ　ＯＣＸのＶａｌｕｅプロパティの変更

このＯＣＸは、Ｖａｌｕｅというプロパティを変更することによって、０からＦまでの１６進数を表示する。そこで、Ｖａｌｕｅプロパティを７にセットする。



ヘ　新しい値の表示

Ｖａｌｕｅプロパティを７にセットした結果が表示される。



(3) ＯＣＸの実現方法

Visual C++には、ＯＣＸを作成するためのＡｐｐＷｉｚａｒｄという機能が装備されている。そのため、比較的容易にＯＣＸが作成できる。

しかし、そのためには、Ｃ＋＋の知識が必要なことはいうまでもない。

７ＳｅｇＬＥＤを実現するためには、以下のようなコードを記述する。

```
// EP7SegLEDCtl.cpp : CEP7SegLEDCtrl OLE コントロール クラスのインプリメンテーション

#include "stdafx.h"
#include "EP7SegLED.h"
#include "EP7SegLEDCtl.h"
#include "EP7SegLEDPpg.h"


#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


IMPLEMENT_DYNCREATE(CEP7SegLEDCtrl, COleControl)


////////////////////////////////////////////////////////////////////////////
// メッセージ マップ

BEGIN_MESSAGE_MAP(CEP7SegLEDCtrl, COleControl)
    //{{AFX_MSG_MAP(CEP7SegLEDCtrl)
    // メモ・ClassWizard はメッセージ マップのエントリを追加又は削除します
    //    この位置に生成されるコードを編集しないでください！
    //}}AFX_MSG_MAP
    ON_OLEVERB(AFX_IDS_VERB_PROPERTIES, OnProperties)
END_MESSAGE_MAP()


////////////////////////////////////////////////////////////////////////////
// ディスパッチ マップ

BEGIN_DISPATCH_MAP(CEP7SegLEDCtrl, COleControl)
    //{{AFX_DISPATCH_MAP(CEP7SegLEDCtrl)
    DISP_PROPERTY_NOTIFY(CEP7SegLEDCtrl, "Value", m_value, OnValueChanged, VT_I
2)
    DISP_PROPERTY_NOTIFY(CEP7SegLEDCtrl, "Thick", m_thick, OnThickChanged, VT_I
2)
    DISP_STOCKFUNC_REFRESH()
    DISP_STOCKPROP_BORDERSTYLE()
    DISP_STOCKPROP_ENABLED()
    DISP_STOCKPROP_BACKCOLOR()
    DISP_STOCKPROP_FORECOLOR()
    //}}AFX_DISPATCH_MAP
    DISP_FUNCTION_ID(CEP7SegLEDCtrl, "AboutBox", DISPID_ABOUTBOX, AboutBox, VT_E
MPTY,
VTS_NONE)
END_DISPATCH_MAP()
```

```
////////////////////////////////////////////////////////////////////////////
// イベント マップ

BEGIN_EVENT_MAP(CEP7SegLEDCtrl, COleControl)
    //{{AFX_EVENT_MAP(CEP7SegLEDCtrl)
    EVENT_STOCK_CLICK()
    EVENT_STOCK_DBLCLICK()
    EVENT_STOCK_KEYDOWN()
    EVENT_STOCK_KEYPRESS()
    EVENT_STOCK_KEYUP()
    EVENT_STOCK_MOUSEDOWN()
    EVENT_STOCK_MOUSEMOVE()
    EVENT_STOCK_MOUSEUP()
    //}}AFX_EVENT_MAP
END_EVENT_MAP()


////////////////////////////////////////////////////////////////////////////
// プロパティ ページ

// TODO: プロパティ ページを追加して、BEGIN 行の最後にあるカウントを増やしてくだ
い。
BEGIN_PROPPAGEIDS(CEP7SegLEDCtrl, 1)
    PROPPAGEID(CEP7SegLEDPropPage::guid)
END_PROPPAGEIDS(CEP7SegLEDCtrl)


////////////////////////////////////////////////////////////////////////////
// クラス ファクトリと guid を初期化します

IMPLEMENT_OLECREATE_EX(CEP7SegLEDCtrl, "EP7SEGLED.EP7SegLEDCtrl.1",
    0xb83aaf23, 0x931e, 0x11d0, 0x9d, 0xe5, 0x44, 0x45, 0x53, 0x54, 0, 0)


////////////////////////////////////////////////////////////////////////////
// タイプ ライブラリ ID とバージョン

IMPLEMENT_OLETYPELIB(CEP7SegLEDCtrl, _tlid, _wVerMajor, _wVerMinor)


////////////////////////////////////////////////////////////////////////////
// インターフェイス ID

const IID BASED_CODE IID_DEP7SegLED =
            { 0xb83aaf21, 0x931e, 0x11d0, { 0x9d, 0xe5, 0x44, 0x45, 0x53, 0x54,
0, 0 } };
const IID BASED_CODE IID_DEP7SegLEDEvents =
            { 0xb83aaf22, 0x931e, 0x11d0, { 0x9d, 0xe5, 0x44, 0x45, 0x53, 0x54,
0, 0 } };


////////////////////////////////////////////////////////////////////////////
// タイプ情報の制御
```

```
static const DWORD BASED_CODE _dwEP7SegLEDOleMisc =
    OLEMISC_ACTIVATEWHENVISIBLE |
    OLEMISC_SETCLIENTSITEFIRST |
    OLEMISC_INSIDEOUT |
    OLEMISC_CANTLINKINSIDE |
    OLEMISC_RECOMPOSEONRESIZE;

IMPLEMENT_OLECTLTYPE(CEP7SegLEDCtrl, IDS_EP7SEGLED, _dwEP7SegLEDOleMisc)


////////////////////////////////////////////////////////////////////////
// CEP7SegLEDCtrl::CEP7SegLEDCtrlFactory::UpdateRegistry ・
// CEP7SegLEDCtrl のシステム レジストリのエントリを追加又は削除します

BOOL CEP7SegLEDCtrl::CEP7SegLEDCtrlFactory::UpdateRegistry(BOOL bRegister)
{
    if (bRegister)
            return AfxOleRegisterControlClass(
                    AfxGetInstanceHandle(),
                    m_clsid,
                    m_lpszProgID,
                    IDS_EP7SEGLED,
                    IDB_EP7SEGLED,
                    FALSE,                          //  挿入不可能
                    _dwEP7SegLEDOleMisc,
                    _tlid,
                    _wVerMajor,
                    _wVerMinor);
    else
            return AfxOleUnregisterClass(m_clsid, m_lpszProgID);
}


////////////////////////////////////////////////////////////////////////
// CEP7SegLEDCtrl::CEP7SegLEDCtrl ・ コンストラクタ

CEP7SegLEDCtrl::CEP7SegLEDCtrl()
{
    InitializeIIDs(&IID_DEP7SegLED, &IID_DEP7SegLEDEvents);

    // TODO: この位置にコントロールのインスタンス データの初期化処理を追加してく
ださい
    SetInitialSize(82,120);
}


////////////////////////////////////////////////////////////////////////
// CEP7SegLEDCtrl::-CEP7SegLEDCtrl ・ デストラクタ

CEP7SegLEDCtrl::-CEP7SegLEDCtrl()
{
    // TODO: この位置にコントロールのインスタンス データの後処理用のコードを追加
てください
```

```
}


////////////////////////////////////////////////////////////////////////////
// CEP7SegLEDCtrl::OnDraw ・ 描画関数

void CEP7SegLEDCtrl::OnDraw(
                    CDC* pdc, const CRect& rcBounds, const CRect& rcInvalid)
{
    // TODO: 以下のコードを描画用のコードに置き換えてください
    PrintSegment(pdc,rcBounds);
}


////////////////////////////////////////////////////////////////////////////
// CEP7SegLEDCtrl::DoPropExchange ・ 永続性のサポート

void CEP7SegLEDCtrl::DoPropExchange(CPropExchange* pPX)
{
    ExchangeVersion(pPX, MAKELONG(_wVerMinor, _wVerMajor));
    COleControl::DoPropExchange(pPX);

    // TODO: 継続表示属性を持つ各カスタム プロパティ用の PX_ 関数の呼び出しを追
加
てください
    PX_Short(pPX,_T("Value"),m_value,0);
    PX_Short(pPX,_T("Thick"),m_thick,5);
}


////////////////////////////////////////////////////////////////////////////
// CEP7SegLEDCtrl::OnResetState ・ コントロールのプロパティ値をリセット

void CEP7SegLEDCtrl::OnResetState()
{
    COleControl::OnResetState();  // DoPropExchange を呼び出してデフォルト値にリ
セット

    // TODO: この位置にコントロールの状態をリセットする処理を追加してください
    m_thick=5;
    m_value=0;
}


////////////////////////////////////////////////////////////////////////////
// CEP7SegLEDCtrl::AboutBox ・ "バージョン情報" のダイアログ ボックスを表示

void CEP7SegLEDCtrl::AboutBox()
{
    CDialog dlgAbout(IDD_ABOUTBOX_EP7SEGLED);
    dlgAbout.DoModal();
}
```

```
/////////////////////////////////////////////////////////////////////////////
// CEP7SegLEDCtrl メッセージ ハンドラ


void CEP7SegLEDCtrl::OnValueChanged()
{
    // TODO: ﾊﾝﾄﾞﾗ ｺｰﾄﾞを追加してください
    CRect rcBounds;
    CClientDC dc(this);
    GetClientRect(&rcBounds);
    PrintSegment(&dc,&rcBounds);
    SetModifiedFlag();
}

void CEP7SegLEDCtrl::PrintSegment(CDC* pdc, const CRect& rcBounds)
{
    long lX,lY;
    BOOL bSegment[9];
    // START PROCESS
    pdc->SetBkMode(TRANSPARENT);
    CBrush bkBrush(TranslateColor(GetBackColor()));
    pdc->FillRect(rcBounds,&bkBrush);
    HPEN hNewPen=CreatePen(PS_SOLID,m_thick,TranslateColor(GetForeColor()));
    HPEN hOldPen=(HPEN)SelectObject(pdc->m_hDC,hNewPen);
    switch(m_value)
    {
    case 0:
            bSegment[0]=bSegment[1]=bSegment[2]=bSegment[3]=bSegment[4]=bSegment
[5]=TRUE;
            bSegment[6]=bSegment[7]=bSegment[8]=FALSE;
            break;
    case 1:
            bSegment[1]=bSegment[2]=TRUE;

bSegment[0]=bSegment[3]=bSegment[4]=bSegment[5]=bSegment[6]=bSegment[7]=bSegment
[8]=FALSE;
            break;
    case 2:
            bSegment[0]=bSegment[1]=bSegment[3]=bSegment[4]=bSegment[6]=TRUE;
            bSegment[2]=bSegment[5]=bSegment[7]=bSegment[8]=FALSE;
            break;
    case 3:
            bSegment[0]=bSegment[1]=bSegment[2]=bSegment[3]=bSegment[6]=TRUE;
            bSegment[4]=bSegment[5]=bSegment[7]=bSegment[8]=FALSE;
            break;
    case 4:
            bSegment[1]=bSegment[2]=bSegment[5]=bSegment[6]=TRUE;
            bSegment[0]=bSegment[3]=bSegment[4]=bSegment[7]=bSegment[8]=FALSE;
            break;
    case 5:
            bSegment[0]=bSegment[2]=bSegment[3]=bSegment[5]=bSegment[6]=TRUE;
            bSegment[1]=bSegment[4]=bSegment[7]=bSegment[8]=FALSE;
            break;
    case 6:
```

```
            bSegment[0]=bSegment[2]=bSegment[3]=bSegment[4]=bSegment[5]=
            bSegment[6]=TRUE;
            bSegment[1]=bSegment[7]=bSegment[8]=FALSE;
            break;
    case 7:
            bSegment[0]=bSegment[1]=bSegment[2]=TRUE;

bSegment[3]=bSegment[4]=bSegment[5]=bSegment[6]=bSegment[7]=bSegment[8]=FALSE;
            break;
    case 8:

bSegment[0]=bSegment[1]=bSegment[2]=bSegment[3]=bSegment[4]=bSegment[5]=bSegment
[6]=TRUE;
            bSegment[7]=bSegment[8]=FALSE;
            break;
    case 9:
            bSegment[0]=bSegment[1]=bSegment[2]=bSegment[3]=bSegment[5]=bSegment
[6]=TRUE;
            bSegment[4]=bSegment[7]=bSegment[8]=FALSE;
            break;
    case 10:
            bSegment[0]=bSegment[1]=bSegment[2]=bSegment[4]=bSegment[5]=bSegment
[6]=TRUE;
            bSegment[3]=bSegment[7]=bSegment[8]=FALSE;
            break;
    case 11:
            bSegment[2]=bSegment[3]=bSegment[4]=bSegment[5]=bSegment[6]=TRUE;
            bSegment[0]=bSegment[1]=bSegment[7]=bSegment[8]=FALSE;
            break;
    case 12:
            bSegment[0]=bSegment[3]=bSegment[4]=bSegment[5]=TRUE;
            bSegment[1]=bSegment[2]=bSegment[6]=bSegment[7]=bSegment[8]=FALSE;
            break;
    case 13:
            bSegment[1]=bSegment[2]=bSegment[3]=bSegment[4]=bSegment[6]=TRUE;
            bSegment[0]=bSegment[5]=bSegment[7]=bSegment[8]=FALSE;
            break;
    case 14:
            bSegment[0]=bSegment[3]=bSegment[4]=bSegment[5]=bSegment[6]=TRUE;
            bSegment[1]=bSegment[2]=bSegment[7]=bSegment[8]=FALSE;
            break;
    case 15:
            bSegment[0]=bSegment[4]=bSegment[5]=bSegment[6]=TRUE;
            bSegment[1]=bSegment[2]=bSegment[3]=bSegment[7]=bSegment[8]=FALSE;
            break;
    }
    // No.0
    if(bSegment[0])
    {
            lX=rcBounds.left+(rcBounds.right·rcBounds.left)*31/82;
            lY=rcBounds.top+(rcBounds.bottom·rcBounds.top)*5/24;
            MoveToEx(pdc·>m_hDC, (int)lX, (int)lY, NULL);
            lX=rcBounds.left+(rcBounds.right·rcBounds.left)*58/82;
            LineTo(pdc·>m_hDC, (int)lX, (int)lY);
```

```
}
// No.1
if(bSegment[1])
{
        lX=rcBounds.left+(rcBounds.right-rcBounds.left)*64/82;
        lY=rcBounds.top+(rcBounds.bottom-rcBounds.top)*6/24;
        MoveToEx(pdc->m_hDC, (int)lX, (int)lY,NULL);
        lX=rcBounds.left+(rcBounds.right-rcBounds.left)*61/82;
        lY=rcBounds.top+(rcBounds.bottom-rcBounds.top)*11/24;
        LineTo(pdc->m_hDC, (int)lX, (int)lY);
}
// No.2
if(bSegment[2])
{
        lX=rcBounds.left+(rcBounds.right-rcBounds.left)*60/82;
        lY=rcBounds.top+(rcBounds.bottom-rcBounds.top)*13/24;
        MoveToEx(pdc->m_hDC, (int)lX, (int)lY,NULL);
        lX=rcBounds.left+(rcBounds.right-rcBounds.left)*57/82;
        lY=rcBounds.top+(rcBounds.bottom-rcBounds.top)*18/24;
        LineTo(pdc->m_hDC, (int)lX, (int)lY);
}
// No.3
if(bSegment[3])
{
        lX=rcBounds.left+(rcBounds.right-rcBounds.left)*24/82;
        lY=rcBounds.top+(rcBounds.bottom-rcBounds.top)*19/24;
        MoveToEx(pdc->m_hDC, (int)lX, (int)lY,NULL);
        lX=rcBounds.left+(rcBounds.right-rcBounds.left)*51/82;
        LineTo(pdc->m_hDC, (int)lX, (int)lY);
}
// No.4
if(bSegment[4])
{
        lX=rcBounds.left+(rcBounds.right-rcBounds.left)*21/82;
        lY=rcBounds.top+(rcBounds.bottom-rcBounds.top)*13/24;
        MoveToEx(pdc->m_hDC, (int)lX, (int)lY,NULL);
        lX=rcBounds.left+(rcBounds.right-rcBounds.left)*18/82;
        lY=rcBounds.top+(rcBounds.bottom-rcBounds.top)*18/24;
        LineTo(pdc->m_hDC, (int)lX, (int)lY);
}
// No.5
if(bSegment[5])
{
        lX=rcBounds.left+(rcBounds.right-rcBounds.left)*25/82;
        lY=rcBounds.top+(rcBounds.bottom-rcBounds.top)*6/24;
        MoveToEx(pdc->m_hDC, (int)lX, (int)lY,NULL);
        lX=rcBounds.left+(rcBounds.right-rcBounds.left)*22/82;
        lY=rcBounds.top+(rcBounds.bottom-rcBounds.top)*11/24;
        LineTo(pdc->m_hDC, (int)lX, (int)lY);
}
// No.6
if(bSegment[6])
{
        lX=rcBounds.left+(rcBounds.right-rcBounds.left)*27/82;
```

```
            lY=rcBounds.top+(rcBounds.bottom-rcBounds.top)*12/24;
            MoveToEx(pdc->m_hDC,(int)lX,(int)lY,NULL);
            lX=rcBounds.left+(rcBounds.right-rcBounds.left)*54/82;
            LineTo(pdc->m_hDC,(int)lX,(int)lY);
    }
    // No.7
    if(bSegment[7])
    {
            lX=rcBounds.left+(rcBounds.right-rcBounds.left)*9/82;
            lY=rcBounds.top+(rcBounds.bottom-rcBounds.top)*19/24;
            MoveToEx(pdc->m_hDC,(int)lX,(int)lY,NULL);
            lY=rcBounds.top+(rcBounds.bottom-rcBounds.top)*19/24+1;
            LineTo(pdc->m_hDC,(int)lX,(int)lY);
    }
    // No.8
    if(bSegment[8])
    {
            lX=rcBounds.left+(rcBounds.right-rcBounds.left)*69/82;
            lY=rcBounds.top+(rcBounds.bottom-rcBounds.top)*19/24;
            MoveToEx(pdc->m_hDC,(int)lX,(int)lY,NULL);
            lY=rcBounds.top+(rcBounds.bottom-rcBounds.top)*19/24+1;
            LineTo(pdc->m_hDC,(int)lX,(int)lY);
    }
    // END PROCESS
    SelectObject(pdc->m_hDC,hOldPen);
    DeleteObject(hNewPen);
}

void CEP7SegLEDCtrl::OnThickChanged()
{
    // TODO: ﾊﾝﾄﾞﾗ ｺｰﾄﾞを追加してください

    SetModifiedFlag();
}
```

5 プログラミングの一般的な考え方

今回開発した訓練計画システムのような小規模なソフトウェアでも、オブジェクト思考の考え方や、ＯＣＸによる部品の再利用の考え方は非常に効果的である。他人の作成してくれた部品を活用するというＯＣＸの考え方は、基本的にはオブジェクト思考の考え方と相通じるものがある。

そこで、今後Visual Basicでシステムを開発する場合には、以下の２点に留意すればよいのではないかと考える。

第１は、既存のＯＣＸをなるべく活用するということである。ＯＣＸを導入した方が、ソフトウェアの生産性は向上する。また、既存のＯＣＸで満足できないときも、既存のＯＣＸと、他の部品を組み合わせることにより、満足できる状況を生み出せることも多い。

第２は、オブジェクト思考の考え方をなるべく取り入れて、再利用の可能性がありそうなところでは、クラスを作成しておくということである。これまで、オブジェクト思考の考え方と無縁でいられたVisual Basicでのプログラミングの世界においても、今後、オブジェク

ト思考の考え方が必要とされていくものと考えられる。そのときに備えて、クラスの作成法を習得しておくことが大切である。

6 実装していない項目

今回の開発では、以下のような項目について、実装を行っていない。
これらは、更に訓練計画システムを使いやすくするために、改良すべき点である。

(1) メニューの導入

Ｗｉｎｄｏｗｓのソフトウェアは、それをはじめて使う人でも容易に操作できるよう、ＧＵＩのガイドラインに従って画面を開発するのが通例である。

例えば、ソフトウェアには、メニューを設け、メニューの項目は、左端から

- ファイル
- 編集
- ヘルプ

をならべるといったガイドラインである。

これに従うことによって、はじめてソフトウェアを見たユーザでも、使い方の推測がつくようになる。

しかし、今回開発した訓練計画システムは、そのガイドラインをまったく無視して作成している。それは、これまでＤＯＳのプログラムしか使用したことのないユーザも、ボタンを押すだけで、簡単に使用できるようにするためである。

そこで、今後、訓練計画システムにメニューを導入し、使いやすくすることを考える必要がある。

(2) バックアップ機能の追加

現在、訓練計画システムには、バックアップをする機能が存在しない。そのため、ユーザは自らの手で、訓練開発システムが変更するファイルを手動でバックアップしなければならない。

そこで、今後、自動バックアップを行う機能を追加し、データの保全を図る必要がある。

(3) 機密保護機能

現在、訓練開発システムには、機密保護機能が存在しない。そのため、ユーザの保管したファイルは、他の人が自由に改変することができる。

そこで、今後、機密保護機能を追加し、データの安全性を高める必要がある。