Ｖｉｓｕａｌ　Ｂａｓｉｃ

（№３）

科　　名		氏　　名	
Tutorial group			

2004.8

例題 6 次の機能の説明を読んで、キーボードから入力した文字列を、順編成ファイルへ書き込むアプリケーションを作成しなさい。


D:¥Home¥User1XX¥VB¥Reidai6

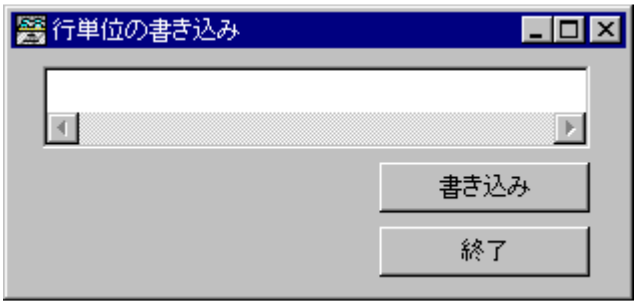

Project1(Writeln.vbp)


Form1(Writeln.frm)

プロジェクトの設定項目

スタートアップの設定	Form1
タイトル	例題 6 行単位の書き込み
アイコン	FILES06.ICO
Picture のセットアップされている path	
¥¥OkaGp200¥Public¥Microsoft Visual Basic¥GRAPHICS¥ICONS¥OFFICE	

〔アプリケーションの外観〕



【機能】

(1) CommandButton コントロール(書き込み)を Click すると、TextBox コントロールの文字列を次の**順編成ファイル**に１行書き込む。
順編成ファイルの仕様は、次のとおりである。

ファイルアクセス方法	シーケンシャルアクセス
レコード長	可変長
ファイル名	D:¥Home¥User1XX¥VB¥Reidai6¥Rei-6-1.TXT

(2) **順編成ファイル**に書き込んだ後、カーソルを自動的に TextBox コントロールに移動する。

(3) 前述(2)の操作後、TextBox コントロールの文字列を消去する。

(4) CommandButton コントロール(終了)を Click すると、アプリケーションは終了する。

(5)  を Click してアプリケーションを終了させても、ファイルを正しくクローズする。

【解説】

(1) **順編成ファイル**への書き込みは、**Print#**又は**Write#ステートメント**を使用する。

【構文】

Print #*filenumber*,[*outputlist*]

filenumber　　… ファイル番号を指定。

output/ist　　… 出力する式、数式又は文字列を指定。

解説：**Print#ステートメント**は、*output/ist* に記述されたとおりのデータのイメージをファイルに書き込む。このファイルは、**Line input#ステートメント**で読み込むことができる。

また、*output/ist* に複数の式、数式又は文字列を記述する場合、**区切り記号(delimiter/separator)**で区切って記述する。

指定項目	内容
Spc(*n*)	***n***個のスペースを出力する。
Tab(*n*)	レコードの先頭を1として、***n***番目の桁位置に次の文字を出力する。 既に出力されている桁位置よりも、小さい桁位置***n***を指定すると自動的に改行されて、次の行の***n***番目の桁位置に次の文字を出力する
区切り記号	
スペース	下記の";"と同じ働きをする。
;	前の文字列に続いて、次の文字列を出力する。
,	出力される次の文字列は14桁区切りで出力する。

※ *output/ist*に記述した式、数式又は文字列を数値として出力する場合、出力される文字コード列の前後に各1個ずつのスペースが挿入される。

※ *output/ist*に記述した式、数式又は文字列を文字列として出力する場合、出力される文字コード列のみが出力される。

例）Print #1, 123; 456　⇒　_123__456_

　　Print #1, "123"; "456"　⇒　123456

【構文】

Write #*filenumber*,[*output/ist*]

filenumber　　… ファイル番号を指定。

output/ist　　… 出力する式、数式又は文字列を指定。

解説：**Write#ステートメント**は、*output/ist* に記述されたデータ項目(*field*)の間に","を挿入してファイルに書き込む。このファイルは、**CSV(Comma Separated Value)**形式ファイルと呼ばれ、**Input#ステートメント**で読み込むことができる。

また、*output/ist* に複数の式、数式又は文字列を記述する場合、**区切り記号(delimiter/separator)**で区切って記述する。

指定項目	内容
Spc(*n*)	***n***個のスペースを出力する。
Tab(*n*)	レコードの先頭を1として、***n***番目の桁位置に次の文字を出力する。 既に出力されている桁位置よりも、小さい桁位置***n***を指定すると自動的に改行されて、次の行の***n***番目の桁位置に次の文字を出力する
区切り記号	
スペース	下記の";"と同じ働きをする。
;	下記の", "と同じ働きをする。
,	出力されるデータ項目の間に","を挿入する。

※ *output/ist*に記述した式、数式又は文字列を数値として出力する場合、出力さ

れる文字コード列のみが出力される。

※ *outputlist* に記述した式、数式又は文字列を文字列として出力する場合、出力される文字コード列は"(*double quotation*)で囲まれて出力される。

例) Write #1, "123"; 456 ⇨ "123",456

【実装】

(1) TextBox コントロールを、Form1 フォームに配置する。また、TextBox コントロールのプロパティを次のとおり設定する。

・ MultiLine プロパティ　: True

・ ScrollBars プロパティ　: 1 - 水平

(2) CommandButton コントロールを、Form1 フォームに配置し、各プロパティを設定する。

(3) 書き込むファイル名は、指定のとおりとし、**宣言セクション(General Declarations)**に定数として定義する。⇦ **重要事項**

(4) ファイルのオープンは、**Form_Load プロシージャ**で行う。ファイルモードは、テキスト･P146 を参照。

(5) 使用可能なファイル番号は、**FreeFile()関数**で取得する。Visual Basic のﾍﾙﾌﾟ(H)を参照。

(6) CommandButton コントロール(書き込み)の処理は、次のとおりである。

① **Print#ステートメント**により、1 行をファイルに書き込む。

② **SetFocus メソッド**により、TextBox コントロールにフォーカスを与える。

③ TextBox コントロールを初期化する。

(7) CommandButton コントロール(終了)でアプリケーションを終了する時、ファイルをクローズするとともに、MsgBox 関数で次のとおり表示する。

prompt:　"終了します。"

buttons:

title:　"例題６－出力用"

戻り値:　なし

(8) Form が閉じられる時の処理は、**Form_QueryUnload イベントプロシージャ**を使用する。

【追加機能】

(1) アプリケーションを起動してフォームが表示された時、カーソルが自動的に TextBox コントロールに移動される。

(2) TextBox コントロールに文字列を入力して、Enter を押下すると、TextBox コントロールの文字列を**順編成ファイル**に 1 行書き込む。

【課題】

(1) 機能(3)をコーディングしなさい。

(2) 機能(4)と(5)をコーディングしなさい。

(3) **CSV 形式**について、調査しなさい。

(4) このアプリケーションでテキストファイルを作成後、**応用 5-3** のアプリケーションでそのファイルを読み込ませなさい。

(5) アプリケーション中のソースコードの **Print#ステートメント**を、**Write#ステートメント**に変更して動作させなさい(確認後、元に戻しておくこと)。

【考察】

《プログラムの検証》

1. スプラッシュスクリーン(frmSplash)

```
Private Sub Form_Load()
    ' このスプラッシュスクリーンを、ディスプレイの中央に表示する。
① → With Me
        .Top = (Screen.Height - .Height) / 2
        .Left = (Screen.Width - .Width) / 2
    End With
    ' タイマーを起動する。
    Timer1.Enabled = True

End Sub
Private Sub Timer1_Timer()
    ' メインウィンドウをロードする。
    Load frmBrowser
End Sub
```

【構文】

With *object*
    [*statements*]
End With

object　　… オブジェクト名、又はユーザ定義型名を指定。

statements　　… *object* に対して実行するステートメントを記述。

解説：指定した *object* に対して *object* 名を再定義することなく、一連のステートメントを実行することができる。

【考察】

2. メインウィンドウ(frmBrowser)

```
Private Sub Form_Load()
```

```
    ' このメインウィンドウを、ディスプレイの中央に表示する。
    Top = (Screen.Height - Height) / 2
    Left = (Screen.Width - Width) / 2
    ' スプラッシュスクリーンを非表示・アンロードする。
    Unload frmSplash
    ' このフォームを表示する。
    Me.Show
End Sub
Private Sub cmdRead_Click()
    Dim sText As String
    txtStatus = ""                                  '
    GetOpenFileName                                 '
    if txtFileName = "" Then                        '
②→      Exit Sub
    End If
    On Error Goto ErrHandler                        '
    txtStatus = "読込み中"                          '
    txtStatus.Refresh                               '
    FileNumber = FreeFile                           '
    Open txtFileName For Input As #FileNumber       '
    txtReadIn = ""                                  '
③→  Do While Not EOF(FileNumber)
        Line Input #FileNumber, sText               '
        txtReadIn = txtReadIn & sText & Chr(&HD) & Chr(&HA)
    Loop
    Close #FileNumber                               '
    txtStatus = "読込み完了"                        '
    On Error Goto 0                                 '
    Exit Sub
ErrHandler:
    txtStatus = "読込みエラー"
    MsgBox Err.Description & Chr(&HA) & _
          "Error code = " & Err.Number, vbCritical, "ﾌｧｲﾙを開く"
End Sub
Private Sub cmdEnd_Click()
    MsgBox "終了します。", , "応用５－入力用"
    End
End Sub
```

【考察】

《プログラムの検証》

3. 開くﾀﾞｲｱﾛｸﾞ(frmOpenDLG)

```
④ →Public FullPathName As String                    '
    Private Sub Form_Load()
        ' このダイアログを画面中央に表示
        With frmOpenDLG
            .Top = (Screen.Height - .Height) / 2 '
            .Left = (Screen.Width - .Width) / 2  '
        End With
    ⑤ →frmOpenDLG.Show                          ' このフォームの表示
    End Sub
    Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    ⑥ →frmBrowser.frmOpenDLGStatus = False ' ファイルオープンダイアログ無効を設定
    End Sub
    Private Sub cmdOpen_Click()
        Dim Path As String

        If InStr(txtFileName.Text, """") = 0 Then
            Path = Check_Path(File1.Path)    ' パスの末尾の¥を確認
                                             ' ドライブレターの確認
            Path = Check_Drive(Path, txtFileName.Text)
        Else                                 ' "(double quotation)の削除
            Path = Delete_DQuota(txtFileName.Text)
        End If
        txtFileName.Text = Path              ' TextBox の文字列を再設定

  ⑦ → File_Open txtFileName.Text             ' ファイルを開く(決定)
    End Sub
    Private Sub cmdCancel_Click()            ' このダイアログを終了
        Unload Me                            ' アプリケーション終了
    End Sub
    Private Sub Drive1_Change()
        On Error GoTo ErrHandler             ' エラートラッピングの開始
        Dir1.Path = Drive1.Drive             ' ドライブレターの設定
        Exit Sub
    ErrHandler:
        MsgBox "ﾃﾞﾊﾞｲｽが準備されていません。", vbExclamation, "ﾌｧｲﾙを開く"
    ⑧ →Resume Next                          '
    End Sub
    Private Sub Dir1_Change()
        File1.Path = Dir1.Path               ' パスの設定
    End Sub
    Private Sub File1_Click()
        txtFileName.Text = File1.Filename    ' ファイル名の設定
    End Sub
    Private Sub File1_DblClick()
        File1_Click                          ' ファイル名の設定
        cmdOpen_Click                        ' 開くボタン処理
    End Sub
```

```
Private Sub txtFileName_KeyPress(KeyAscii As Integer)
    ' Enter キーの押下確認とテキスト欄の空欄の確認
⑨ →If (KeyAscii = 13) And (txtFileName.Text <> "") Then '
        cmdOpen_Click                   ' 開くボタン処理
    End If
End Sub
Function Check_Path(ByVal Path As String) As String
    ' パスの末尾の¥を確認
    If Right(Path, 1) <> "¥" Then
        Path = Path & "¥"               ' 末尾に¥を付加した Path の再設定
    End If

    Check_Path = Path                   ' 関数値の設定
End Function
Function Check_Drive(ByVal Path As String, ByVal Filename As String) As String
    ' ドライブレターを確認
    If InStr(Filename, ":") = 0 Then
        Path = Path & Filename          ' 絶対パスとファイル名の指定
    Else
        Path = Filename                 ' ファイル名のみの指定
    End If

    Check_Drive = Path                  ' 関数値の設定
End Function
Function Delete_DQuota(ByVal Path As String) As String
    Dim PathLen As Integer
    ' "(double quotation)の削除
    If Left(Path, 1) = """" Then
        PathLen = Len(Path) - 1         ' 文字列の長さを取得
        Path = Right(Path, PathLen)     ' 左端の"を削除
    End If
    If Right(Path, 1) = """" Then
        PathLen = Len(Path) - 1         ' 文字列の長さを取得
        Path = Left(Path, PathLen)      ' 右端の"を削除
    End If

    Delete_DQuota = Path                ' 関数値の設定
End Function
Sub File_Open(ByVal Path As String)
    Dim FileFound As String

    On Error GoTo OpenError             ' エラートラッピングの開始
    FileFound = Dir(Path)               ' ファイルの存在の確認
    If Len(FileFound) = 0 Then
        FullPathName = ""
    Else
        FullPathName = Path
    End If
    Unload Me                           ' このフォームをアンロード
```

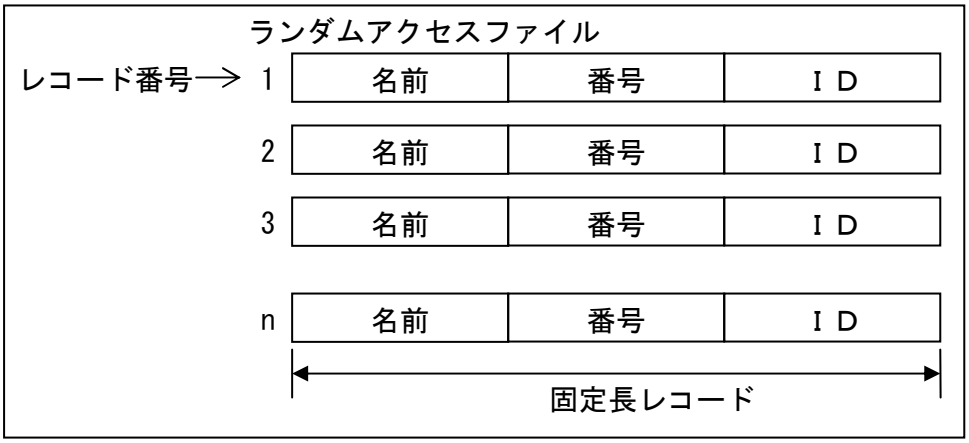
```
        Exit Sub
    OpenError:
        MsgBox Err.Description & Chr(&HA) & "Error Code = " & _
               Err.Number, vbInformation, "ﾌｧｲﾙを開く"        ↑ ⑩
        Resume Next
    End Sub
```

【考察】

5．ランダムアクセスファイル

ランダムアクセスファイルは、任意のレコード位置のデータを読み書きできる。

なお、1件のレコード長は、**固定長**でなければならない。



【*すぐ解決(7)*】

［ファイル名］

D:¥Home¥User1XX¥VB¥BB¥P154(P154.VBP, P154.BAS, P154.FRM)

［プログラムの説明］

テキスト･P154 のプログラムを次のように作成し、その動作を確認しなさい。

［フォーム］

【構文①】－ Open 文

Open *pathname* For Random As #*filenumber* Len=*reclength*

pathname … ファイルのパス及びファイル名を指定する。

filenumber … プログラムが識別する**ファイル番号(1～511)**を指定する。

reclength … ファイルの**レコード長(32,767byte 以下)**を指定する。

解説：Open 文は、ファイルを**ランダムアクセスモード**でオープンする。

ファイルのオープンが成功すると、レコード番号は1番に位置付けられる。

ファイルを**ランダムアクセスモード**でオープンすると、読み込み/書き込み両用モードとなる。

ＶＢでは、**Random キーワード**を省略すると、**ランダムアクセスモード**でファイルをオープンする。

テキスト･P145 及び P146 を参考にすること。

【構文②】－ Put 文

Put #*filenumber*, *recnumber*, *varname*

filenumber … プログラムが識別する**ファイル番号(1～511)**を指定する。

recnumber … ファイル中の書き込みを行う**レコード番号(1～2,147,483,647)**を指定する。
省略すると、ファイルの現在位置に書き込みを行う。

varname … ファイルに書き込む内容を格納している変数を指定する。

解説：**ランダムアクセスファイル**または**バイナリファイル**にデータを書き込む。

【構文③】－ 文字列型

[Dim|Public] *varname* As String … 可変長文字列

[Dim|Public] *varname* As String * *length* … 固定長文字列

varname … 文字列を格納する変数名を記述する。

length … 文字列長を指定する。

解説：ＶＢでは、**可変長文字列**と**固定長文字列**を扱うことができる。
固定長文字列変数に文字を代入した場合、空いた右側は**null(0x00)**で埋められ、指定の長さを越えるとはみ出した部分は捨てられる。

【考察】

[ファイル名]

D:¥Home¥User1XX¥VB¥BB¥P155(P155.VBP, P154.BAS, P155.FRM)

[プログラムの説明]

テキスト･P155 のプログラムを作成し、その動作を確認しなさい。

なお、標準モジュールは"P154.BAS"をそのまま使用する。

【構文④】－ Get 文

Get #*filenumber*, *recnumber*, *varname*

filenumber … プログラムが識別する**ファイル番号(1～511)**を指定する。

recnumber … ファイル中の読み込みを行う**レコード番号(1～2,147,483,647)**を指定する。
省略すると、ファイルの現在位置から読み込みを行う。

varname … ファイルから読み込んだ内容を格納する変数を指定する。

解説：**ランダムアクセスファイル**または**バイナリファイル**からデータを読み込む。

【考察】

［ファイル名］

D:¥Home¥User1XX¥VB¥BB¥P156(P156.VBP, P156.BAS, P156.FRM)

［プログラムの説明］

テキスト･P156 を参考にして、次の仕様のプログラムを作成し、その動作を確認しなさい。

［フォーム］

［仕様］

(1) TextBoxControl(番号)に自然数(1～)を入力し、検索ボタンをクリックすると、ファイルの n レコードに直接位置付けをして、TextBoxControl(コード)に読み込んだコードを、TextBoxControl(氏名)に読み込んだ氏名を表示する。

(2) ファイルの処理は、**ランダムアクセス**による任意のレコードに**直接アクセス**する。

(3) レコード仕様については、先頭からコード：4byte、氏名：20byte で構成され、1 レコードは 24byte ある。

(4) ファイルは、次のディレクトリに保管されているので、各自の指定のディレクトリに保存すること

¥¥XXXXXXXX¥Public¥Microsoft Visual Basic¥P156.DAT

【参考】

① レコード長は、次のように Len 関数によって求めることができる。

Len(ユーザ定義型変数名)

② ファイルサイズは、次のように Lof 関数によって求めることができる。

Lof(ファイル番号)

③ ランダムアクセスファイルのレコード件数は、次のように Lof 関数と Len 関数によって求めることができる。

Lof(ファイル番号) ¥ Len(ユーザ定義型変数名)

【考察】

6．**バイナリファイル**

バイナリファイルは、","や CR-LF などを特別な**コード(機能コード)**として扱わず、**byte 単位**でファイルの先頭から逐次読み込み/書き込みできる。

なお、**バイナリデータ**を格納する変数には、**Byte 型配列**を使用すること。

(**String 型**を使用すると、不要なデータ変換が行われてしまう。)

【*すぐ解決(8)*】

［ファイル名］

D:¥Home¥User1XX¥VB¥BB¥P159(P159.VBP, P159.FRM)

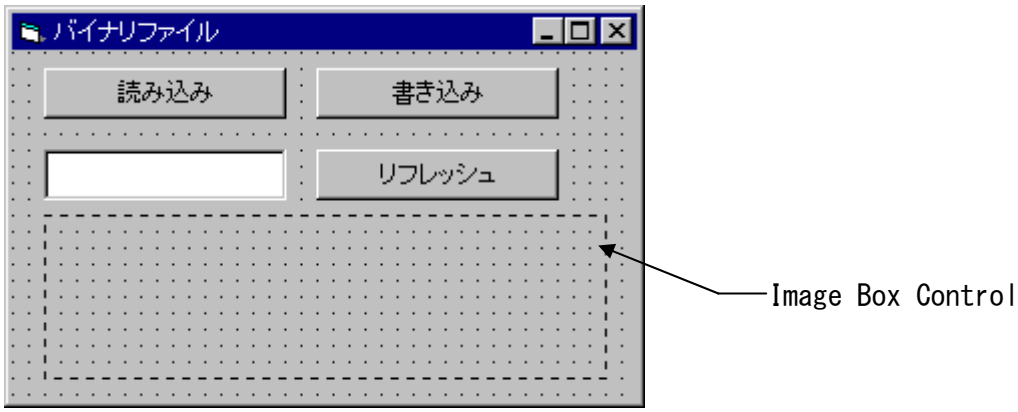
［プログラムの説明］

テキスト･P159 及び P157 のプログラムを次のように作成し、その動作を確認しなさい。

なお、ピクチャーファイルは、次のディレクトリに保管されているので、各自の指定のディレクトリに保存すること。

¥¥XXXXXXXX¥Public¥Microsoft Visual Basic¥HELLO.BMP

［フォーム］



［プログラム］

次のプログラムを、追加コーディングしなさい。

```
Private Sub Command3_Click()                    ' リフレッシュボタンのクリック
    On Error GoTo FileError                     ' エラートラッピングの開始
    Image1.Picture = LoadPicture("D:¥Home¥User1XX¥VB¥TMP¥Hello2.BMP")
    On Error GoTo 0                             ' エラートラッピングの終了
    Image1.Refresh                              ' コントロールを強制的に更新
    ' プロシージャの中止
    Exit Sub
' ラベルの設定
FileError:                                      ' エラーハンドラの設定
    MsgBox "ファイルエラー:" & Err.Description
End Sub
```

【構文⑤】－ LoadPicture 関数

LoadPicture(*pathname*)

pathname　　… ロードするピクチャーのパス及びファイル名を指定する。

解説：Form、Picture 及び Image の各コントロールに *pathname* で指定されたピクチャーをロードする。

pathname を省略あるいは""を指定すると、ピクチャーをクリアする。

なお、ピクチャーとして使用できる主なファイル形式は、次のとおりである。

- **ビットマップ(BMP)、アイコン(ICO)、メタファイル(WMF)**
- **run-length エンコードファイル(RLE)**

【構文⑥】－ Refresh メソッド

*contro*l.Refresh

*contro*l　　… リフレッシュするフォームやコントロールを指定する。

解説：フォームやコントロールを強制的に更新する。

[操作]

P159 及び P157 のプログラムが正しく作成された場合、リフレッシュ ボタンをクリックすると ImageBoxControl にビットマップファイルが表示される。

【考察】

.....

.....

.....

.....

◆ **ランダムアクセス形式**と**バイナリアクセス形式**のまとめ

(1) オープンモード

Random	ファイルが**ランダムアクセスモード**でオープンされる。
Binary	ファイルが**バイナリアクセスモード**でオープンされる。

(2) アクセスステートメント

Put	**ランダムファイル/バイナリファイル**への書き込み。
Get	**ランダムファイル/バイナリファイル**からの読み込み。
Seek	ファイルの現在位置(レコード番号あるいは先頭からの byte 数)を移動する。

(3) ファイルアクセスに関する関数

Seek()	ファイルの現在位置(レコード番号あるいは先頭からの byte 数)を取得する

【考察】

.....

.....

.....

.....

応用 6-1 例題 6 と応用 5-2 を利用して、キーボードから入力した文字列を、任意の順編成ファイルへ書き込むアプリケーションを作成しなさい(応用 5-3 を参考にすること)。

D:¥Home¥User1XX¥VB¥Reidai6

Project1(TextEditor.vbp)

frmSplash(Splash.frm) … Form1

frmWriter(Writer.frm) … Form2

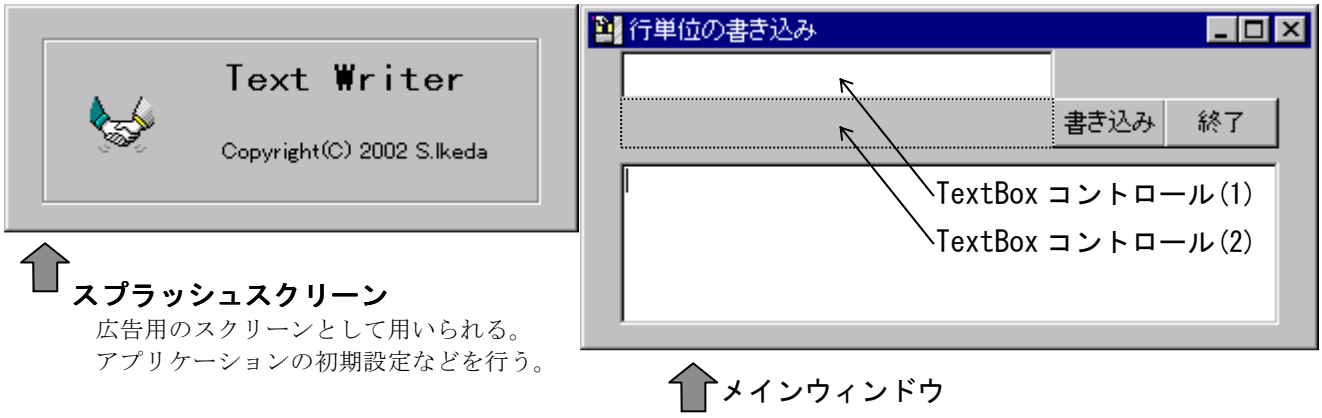
frmOpenDLG(OpenDLG.frm) … Form3

Module1(Writer.bas) … 標準モジュール

プロジェクトの設定項目

スタートアップの設定	frmSplash … Form1
タイトル	行単位の書き込み(応用 6-1)
アイコン	CRDFLE13.ICO
Picture のセットアップされている path	
¥¥OkaGp200¥Public¥Microsoft Visual Basic¥GRAPHICS¥ICONS¥OFFICE	

〔アプリケーションの外観〕



スプラッシュスクリーン
広告用のスクリーンとして用いられる。
アプリケーションの初期設定などを行う。

メインウィンドウ

サンプルアプリケーションが保存されている path とアプリケーション名
¥¥XXXXXXXX¥Public¥Microsoft Visual Basic¥Example¥Apply6-1.EXE

【機能】

(1) アプリケーションが起動された時、画面中央に**スプラッシュスクリーン**を 5 秒間表示する。
(2) **スプラッシュスクリーン**の表示中は、アプリケーションを**タスクバー**に登録しない。
(3) **スプラッシュスクリーン**は、リサイズしない。
(4) **スプラッシュスクリーン**の非表示後、画面中央に**メインウィンドウ**を表示する。
(5) **メインウィンドウ**の CommandButton コントロール(書き込み)を Click すると、**保存ﾀﾞｲｱﾛｸﾞ**を画面中央に表示する(**応用 5-3** を参考)。
(6) **メインウィンドウ**の CommandButton コントロール(終了)を Click すると、アプリケーションは終了する。
(7) **メインウィンドウ**の TextBox コントロール(1)に、書き込んだファイルのパスとファイル名を表示する。

(8) 順編成ファイルに書き込み中は、**メインウィンドウ**の TextBox コントロール(2)に、"**書き込み中**"と表示する。

(9) 順編成ファイルへの書き込みが完了した時、**メインウィンドウ**の TextBox コントロール(2)に、"**書き込み完了**"と表示する。

(10) **保存ﾀﾞｲｱﾛｸﾞ**は、**タスクバー**に登録しない。

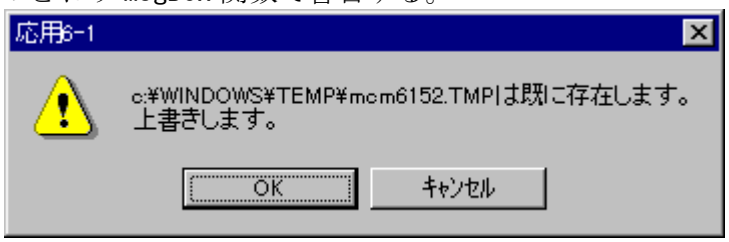
(11) **保存ﾀﾞｲｱﾛｸﾞ**は、リサイズしない。

(12) **保存ﾀﾞｲｱﾛｸﾞ**の機能は、**応用 5-2**･機能(1)～(5)の機能を持つ。

(13) **保存ﾀﾞｲｱﾛｸﾞ**の CommandButton コントロール(キャンセル)を Click すると、ファイル選択処理を中止し、**メインウィンドウ**に戻る。

(14) **保存ﾀﾞｲｱﾛｸﾞ**では、DriveListBox コントロールが操作された時の、実行時のエラー処理を行う。

(15) **保存ﾀﾞｲｱﾛｸﾞ**で、保存するファイルを決定する時、既にそのファイルが存在する場合は、次のとおり MsgBox 関数で警告する。



(16) **メインウィンドウ**のファイルを操作する時の、実行時のエラー処理を行う。

(17) ☒を Click してアプリケーションを終了させても、ファイルを正しくクローズする。

【実装】

(1) **応用 5-2**、**応用 5-3** と **例題 6** で作成した、フォーム、コントロール及びプロシージャを再利用する。

(2) **スプラッシュスクリーン**の実装は、次のとおりである。

① **フォーム及び各コントロールのプロパティは、応用 5-3** の**【実装】**(2)･①～⑥を参考にして設定する。

② フォームを画面中央に表示する操作には、API 関数を使用する。詳細は、サブテキスト･P5～P7 及び**応用 3** の**【実装】**を参考にすること。

③ フォーム及び各コントロールの操作等は、**応用 5-3** の**【実装】**(2)･⑧～⑩を参考にしてイベントプロシージャを記述する。

(3) **メインウィンドウ**の実装は、次のとおりである。

① **例題 6** で実装した各コントロールの配置を、〔アプリケーションの外観〕のとおりに再配置しプロパティを再設定する。

② フォームのプロパティを、次のとおり設定する。

- Icon プロパティ　　：　ICONS¥OFFICE¥FILES09.ICO

③ TextBox コントロール(メイン(*pane*))のプロパティを、次のとおり設定する。

- MultiLine プロパティ　　：　False

④ Form_Load イベントプロシージャでは、次の処理を行う。

- このフォームを、API 関数を使用して、画面中央に表示する一連の処理を行う。
- frmSplash フォームをアンロードする。

・ このフォームを表示する。

⑤ CommandButton コントロール(書き込み)のイベントプロシージャの処理は、次のとおりである。

・ 書き込むファイルのパス及びファイル名を取得するために、frmOpenDLG を起動・表示する。この操作には、Show メソッドを使用する。

【構文】

[*form.*]Show [*style*]

form　　：表示するフォーム名(オブジェクト名)を指定。

style　　：次の表示モードを指定する。

表示モード	*style*		機　能
モードレス	vbModeless	0	フォームを表示後、次のステートメントに制御が移る。
モーダル	vbModal	1	フォームを表示後、表示したフォームが非表示又はアンロードされるまで、次のステートメントに制御を移さない。

解説:フォーム又は MDI フォームを表示する。

・ frmOpenDLG からオープンするファイル名が正しく取得されているかを確認する。
ただし、ファイル名が正しくない場合は MsgBox 関数を表示して、書き込み処理を中止する。

・ **エラートラッピング処理**を開始する。**エラートラッピングルーチン**のプログラムコードは、次のとおりである。

```
ErrHandler:
    txtStatus = "書き込みエラー"  ' 動作Status"書き込みエラー"を表示…TextBox(2)
    MsgBox Err.Description & Chr(&HA) & _
           "Error code = " & Err.Number, vbCritical, "ﾌｧｲﾙを開く"
```

・ TextBox コントロール(2)の Text プロパティに、**"書き込み中"**を設定する。

・ フォーム又はコントロールを強制的に更新する場合、Refresh **メソッド**を使用する。

【構文】

***object*.Refresh**

object　　：オブジェクト名を指定。

解説：フォームやコントロールは、通常イベントの処理が終了した時に自動的に更新されるが、Refresh メソッドにより必要に応じて更新が可能。

・ 使用可能なファイル番号は、FreeFile **関数**で取得する。詳細は、テキスト・P146 を参照。

・ 書き込むファイルを出力モードでオープンする。

・ **Print#ステートメント**により、1 行をファイルに書き込む。

・ オープンしたファイルをクローズする。

・ TextBox コントロール(2)の Text プロパティに、**"書き込み完了"**を設定する。

⑥ このモジュールの**宣言セクション(General Declaration)**では、次のプログラムコードを記述する。

```
Option Explicit
Dim FileNumber As Integer
```

(4) **保存ﾀﾞｲｱﾛｸﾞ**の実装は、次のとおりに変更する。

① Form_Load イベントプロシージャでは、次の処理を行う。

- このフォームを、API 関数を使用して、画面中央に表示する一連の処理を行う。
- このフォームを表示する。

② CommandButton コントロール(保 存)を Click した時のイベントプロシージャに、次の処理を追加する。

- 保存するファイルを決定する時、既にそのファイルが存在する場合は、MsgBox 関数で警告する。この MsgBox 関数の仕様は、次のとおりである。
  (a) *prompt* には、例のとおりに 1 行目に既存のファイルの**絶対パス名**とファイル名を表示する。2 行目に"**上書きします。**"と表示する。
  (b) *buttons* には、  の表示と OK Cancel ボタンを表示する。
  (c) OK 又は Cancel のいずれかのボタンが Click されたかの判定は、MsgBox 関数の値を調査する。

③ 保存するファイルと決定されたファイルのパス及びファイル名は、frmOpenDLG のパブリックスコープ変数 FullPathName に格納する。

- 有効なパス及びファイル名を選択した場合
  決定されたファイルの絶対パスとファイル名を格納する。
- 無効なパス及びファイル名を選択した場合あるいは キャンセル を Click した場合 ""を格納する。

【考察】

【課題】

(1) CommandPrompt を起動して、TYPE コマンドと DUMP.EXE により書き込んだファイルの内容を確認しなさい。

(2) 書き込むファイルを追加モードでオープンして実行し、前述(1)のとおり確認しなさい(確認後、元に戻しておくこと)。

(3) **メインウィンドウ**をリサイズ操作をして、その様子を確認しなさい。

(4) TextBox コントロール(メイン(*pane*))のプロパティを、次のとおりに変更して実行し、前述(1)のとおり確認しなさい(確認後、元に戻しておくこと)。

- MultiLine プロパティ　　: True
- ScrollBars プロパティ　　: 3 - **両方**

(5) TextBox コントロール(メイン(*pane*))内で右 Click して、編集機能が使用可能か確認しなさい。

(6) frmOpenDLG を起動・表示する Show メソッドの引数を省略して実行し、動作を確認しなさい。

【考察】

応用 6-2 応用 6-1 を次のとおりに変更しなさい。

【機能】

・ **メインウィンドウ**をリサイズ操作した場合、各コントロールの大きさや配置位置を調整する。

【解説】

Visual Basic では、画面の座標の単位にデフォルトで **twip** を使用する。1cm が 567twip、1ｲﾝﾁが 1440twip に相当する。なお、ScaleMode **プロパティ**を変更すると、座標の単位が変更される。

ScaleMode プロパティ

0	ユーザ定義の単位	4	キャラクタ(水平:120twip/垂直:240twip)
1	twip(デフォルト)	5	インチ
2	ポイント(72ﾎﾟｲﾝﾄ/ｲﾝﾁ)	6	ｍｍ
3	ピクセル	7	ｃｍ

Visual Basic で、オブジェクトの大きさや配置位置を設定・取得するプロパティは、次のとおりである。

Top	コントロールのコンテナ座標における上端座標(*y*)を設定・取得する。
Left	コントロールのコンテナ座標における左端座標(*x*)を設定・取得する。
Height	コントロールの外形の高さを設定・取得する。
Width	コントロールの外形の幅を設定・取得する。

※ コンテナ座標 … あるオブジェクト(コントロール)が載っている外のオブジェクトの座標。

【実装】

(1) リサイズ操作に合わせて大きさを調整するコントロールを、ピックアップする。

・ リサイズすることにより、表示される情報量が変化した方が良い、又は変化しても構わないコントロールに焦点を当てる。

(例) PictureBox, TextBox, ComboBox, ListBox, ScrollBar, Image など

【Memo】

(2) リサイズ操作に合わせて配置位置を調整するコントロールを、ピックアップする。

・ リサイズしても、表示される情報量が変化しない、又は変化すると不都合が生じるコントロールに焦点を当てる。

(例) Label, Frame, CommandButton, CheckBox, OptionButton など

【Memo】

(3) 大きさを調整するコントロールの、相対的な大きさや間隔を決める。

【Memo】

(4) 配置位置を調整するコントロールの、相対的な配置位置を決める。

【Memo】

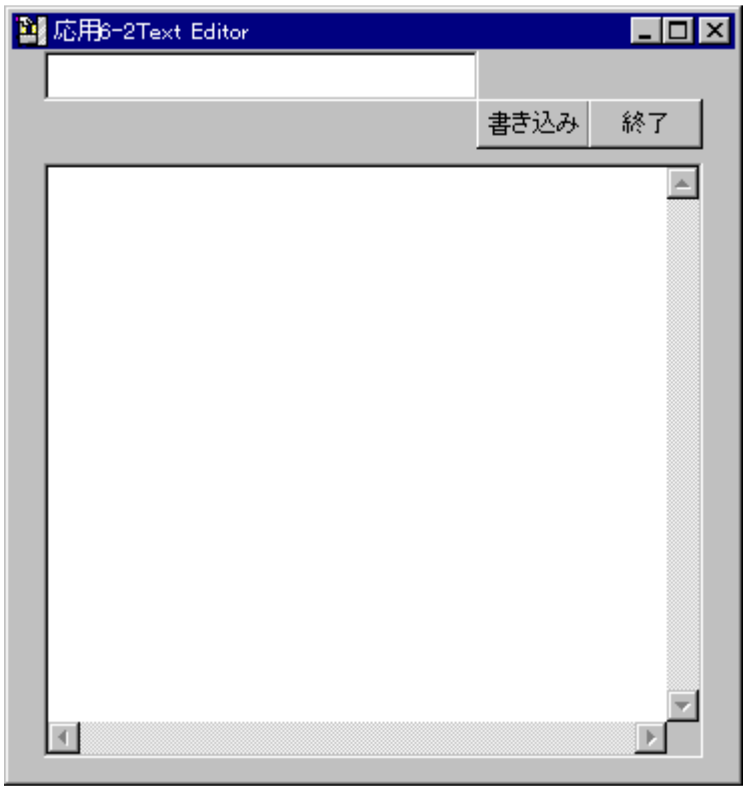
(5) **Form_Resize イベントプロシージャ**に、コントロールの大きさや配置位置を調整するプログラムコードを記述する(テキスト･P124～P125 を参照)。

サンプルアプリケーションが保存されている path とアプリケーション名

¥¥XXXXXXXX¥Public¥Microsoft Visual Basic¥Example¥Apply6-2.EXE

【考察】

〔アプリケーションの外観〕



応用 6-3 応用 6-2 に、次のとおりに機能を追加しなさい。

【機能】

・ **メインウィンドウ**のテキストペインのフォントを、任意のフォントに変更できる。

【解説】

Visual Basic では、Windows に共通のダイアログボックスを提供している。この機能は、**コモンダイアログボックスコントロール(Common Dialogue Box Control)**に実装されている。

コモンダイアログボックスコントロールには、複数のダイアログボックスが実装されているため、使用するダイアログボックスは、次のメソッドを実行することで、それぞれのダイアログボックスが表示される。

ダイアログボックスの種類とメソッド

メソッド	表示されるダイアログボックスの種類
ShowColor	「**色の設定**」ダイアログボックスを表示。
ShowFont	「**フォントの指定**」ダイアログボックスを表示。
ShowHelp	ヘルプエンジンの起動。
ShowPrinter	「**印刷**」ダイアログボックスを表示。
ShowOpen	「**ファイルを開く**」ダイアログボックスを表示。
ShowSave	「**名前を付けて保存**」ダイアログボックスを表示。

また、**コモンダイアログボックスコントロール**は次のディレクトリに格納されている。

Windows NT 系 … Winnt¥System32

Windows 9X 系 … Windows¥System

各ダイアログボックスは、操作しやすいようなユーザインタフェースを提供するだけであるから、ユーザが選択した設定値などはダイアログボックスの各プロパティに格納されている。

したがって、これらの操作した結果をオブジェクトに反映させるには、ダイアログボックスのプロパティ値を操作対象となるオブジェクトのプロパティに設定する必要がある。

(例) txtDsp.Font.Name = cdlgEx3.**FontName** … 選択した**フォント名**(F):の値を、オブジェクトのフォント情報プロパティに設定。

txtDsp.Font.Bold = cdlgEx3.**FontBold** … 選択した**スタイル**(Y):の値を、オブジェクトのフォント情報プロパティに設定。

【実装】

(1) **ツールボックス**上の CommonDialog  を Click して、frmWriter フォームに配置する。
CommonDialog コントロールは、デザイン時には表示されるが、実行時は非表示となる。
また、コントロールのサイズは、自動的にディフォルト値に設定される。

(2) frmWriter フォームに、CommandButton コントロール(ﾌｫﾝﾄ)を配置する(一時的に使用するだけなので、配置場所は任意)。

(3) テキスト･P554 を参考に、**Command1_Click()イベントプロシージャ**に**コモンダイアログボックス**を表示するステートメントを記述する。

(例) CommonDialog1.**Flags** = &H3& **Or** &H100& … **コモンダイアログボックス**のオプションを設定する。

CommonDialog1.**ShowFont** … **コモンダイアログボックス**を表

示する。

(4)「**フォントの指定**」ダイアログボックスのオプション値は、次のとおりである。

オプション値		説　明
cdlCFANSIOnly	&H400&	Windows の文字セットを使うフォントだけを選択。
cdlCFApply	&H200&	**適用ボタン**が使用可能。
cdlCFBoth	&H3&	利用可能な**プリンタフォント**と**スクリーンフォント**を表示。
cdlCFEffects	&H100&	**取り消し線**、**下線**、**色**の設定が可能。
cdlCFFixedPichOnly	&H4000&	**固定ピッチフォント**だけを表示。
cdlCFForceFontExist	&H10000&	存在しないフォントを選択した時、エラーメッセージを表示。
cdlCFHelpButton	&H4&	**ヘルプボタン**を表示。
cdlCFLimitSize	&H2000&	Min、Max プロパティで指定した範囲のサイズだけを表示。
cdlCFNoSimulations	&H1000&	GDI **フォント**のシミュレーションを行わない。
cdlCFNoVectorFonts	&H800&	**ベクタフォント**を選択できないようにする。
cdlCFPrinterFonts	&H2&	プリンタがサポートするフォントだけを表示。
cdlCFScalableOnly	&H20000&	**スケーラブルフォント**だけを表示。
cdlCFScreenFonts	&H1&	システムがサポートする**スクリーンフォント**だけを表示。
cdlCFTTOnly	&H40000&	TrueType **フォント**だけを表示。
cdlCFWYSIWYG	&H8000&	プリンタとスクリーンの両方で使用できるフォントだけを表示

オプションを複数設定する場合は、**Or 演算子**を使用する。

【課題】

(1) 次の用語について調査しなさい。

固定ピットフォント(fixed pitch font/fixed-width font)

等幅フォント(fixed-width font)

プロポーショナルフォント(proportional font)

ビットマップフォント(bit map font)

アウトラインフォント(outline font)

TrueType フォント

PostScript

GDI フォント(Graphic Device Interface font)

デバイスコンテキスト(device context)

(2) 次のプロパティについて、調査しなさい。

プロパティ	説　　明
Color	
FontName	
FontBold	
FontItalic	
FontSize	
FontStrikethru	
FontUnderline	

【考察】

応用 6-4 応用 6-3 に、次のとおりに機能を追加しなさい。

【機能】

・ **メインウィンドウ**のテキストペインを、任意のプリンタに印刷できる。

【解説】

Visual Basic の**「印刷」ダイアログボックス**は、「印刷(P)...」を表示した時に表示される「印刷」ダイアログボックスに相当する。

「印刷」ダイアログボックスでは、プリンタの変更、**印刷範囲**や**印刷部数**などの設定・取得ができる。

「印刷」ダイアログボックスのプロパティ

Copies	印刷部数の設定・取得
FromPage	印刷開始ページの設定・取得
ToPage	印刷終了ページの設定・取得

hDC	選択されたプリンタの**デバイスコンテキスト**の取得
PrinterDefault	システムのディフォルト設定の変更可能を設定

また、PrinterDefault **プロパティ**が True の場合、Printer **オブジェクト**に直接出力することができる。すなわち、Visual Basic における実際の印刷は、Printer **オブジェクト**を使用する。

この Printer オブジェクトにテキストやグラフィックスを描画するには、**グラフィックスメソッド(Circle, Cls, Line, PaintPicture, Point, Print, Pset)**を使用する。その後、EndDoc **メソッド**によって、その出力をプリンタに直接送ることができる。

Printer オブジェクトの主なプロパティ

CurrentX	印刷を行う水平位置を設定。
CurrentY	印刷を行う垂直位置を設定。
DrawWidth	印刷する直線の幅を設定。
FontBold	印刷する文字に太字(**Bold**)を設定。
FontName	印刷する文字のフォント名を設定。
FontSize	印刷する文字のフォントサイズを設定。
Orientation	用紙方向を設定。
PaperSize	用紙サイズを設定。
ScaleMode	印刷位置を表す単位を設定
ScaleHeight	印刷用紙の高さを取得。
TextHeight	印刷する文字の高さを取得。
TextWidth	印刷する文字の幅を取得。

Printer オブジェクトの主なメソッド

EndDoc	現在のページを印刷して印刷を終了し、印刷デバイスに制御を渡す。
KillDoc	印刷ジョブを直ちに終了。
Line	直線データや四角形をプリンタに出力。
NewPage	現在のページを印刷して、印刷用紙を改ページする。
Print	文字データをプリンタに出力。

【実装】

(1) frmWriter フォーム上に、CommonDialog  を確認する(詳細は**応用 6-3** を参照)。

(2) frmWriter フォームに、CommandButton コントロール(印刷)を配置する(一時的に使用するだけなので、配置場所は任意)。

(3) テキスト･P565 を参考に、Command2_Click()イベントプロシージャに**コモンダイアログボックス**を表示するステートメントを記述する。

(例) CommonDialog1.**ShowPrinter** … **コモンダイアログボックス**を表示する。

(4) Visual Basic ヘルプ「Printer オブジェクト」を参考に、Command2_Click()イベントプロシージャにプリンタに出力するステートメントを追加記述する。

(例) **Printer.FontName** = “ＭＳ 明朝” … プリンタに出力するフォントを設定する。
Printer.FontSize = 11 … フォントサイズを設定する。
Printer.Print [*output list*] … 文字データをプリンタに出力する。
Printer.EndDoc … 現在のページを印刷して印刷を終了し、プ

リンタに制御を渡す。

【課題】

(1) **応用 6-2** で設定したフォント及びフォントサイズで印刷できるように変更しなさい。

【Memo】

(2) **「印刷」ダイアログボックス**のオプション値は、次のとおりである。**Flags プロパティ**に設定し、動作を確認しなさい。

オプション値		説　明
cdlPDAllPages	&H0&	「全ﾍﾟｰｼﾞ」ボタンの状態を設定。
cdlPDCollate	&H10&	「部単位で印刷」の状態を設定。
cdlPDDisablePrinterToFile	&H80000&	「ﾌｧｲﾙへ出力」を無効。
cdlPDHideProntToFile	&H100000&	「ﾌｧｲﾙへ出力」を非表示。
cdlPDNoPageNums	&H8&	「ﾍﾟｰｼﾞ指定」「ﾍﾟｰｼﾞから」「ﾍﾟｰｼﾞまで」を無効。
cdlPDNoSelection	&H4&	「選択した部分」を無効。
cdlPDNoWarning	&H80&	通常使うプリンタがない時に、警告メッセージを非表示。
cdlPDPageNums	&H2&	「ﾍﾟｰｼﾞ指定」の状態を設定。
cdlPDPrintSetup	&H40&	「印刷」ダイアログの代わりに「ﾌﾟﾘﾝﾀの設定」ダイアログを表示。
cdlPDPrintToFile	&H20&	「ﾌｧｲﾙへ出力」の状態を設定。
cdlPDReturnDC	&H100&	ダイアログで選択されたプリンタのデバイスコンテキストを取得。
cdlPDReturnIC	&H200&	選択されてプリンタに関する情報コンテキストを取得。
cdlPDReturnDefault	&H400&	通常使うプリンタ名を取得。
cdlPDSelection	&H1&	「選択した部分」の状態を設定。
cdlPDHelpButton	&H800&	ヘルプボタンを表示。
cdlPDUseDevModeCopies	&H40000&	「部数」を無効。

オプションを複数設定する場合は、**Or 演算子**を使用する。

【Memo】

(3) **「印刷」ダイアログボックス**の キャンセル を Click した時、印刷をしないように仕様変更しなさい。

【Memo】

(4) Font **オブジェクト**について、調査しなさい。

【Memo】

(5) Printer **オブジェクト**について、調査しなさい。

【Memo】

【考察】

応用 6-5 応用 6-4 に、次のとおりに機能を追加しなさい。

【機能】

(1) **メインウィンドウ**に **Menu コントロール**を追加し、メニューからも操作できる。

Menu コントロールの各メニュー項目は、次のとおりである。

ﾌｧｲﾙ(F)		編集(E)		表示(V)	
新規作成(N)	Ctrl+N	切り取り(T)	Ctrl+X	ﾌｫﾝﾄ(F)	Ctrl+F
開く(O)...	Ctrl+O	ｺﾋﾟｰ(C)	Ctrl+C		
名前を付けて保存(A)...	Ctrl+A	貼り付け(P)	Ctrl+V		
印刷(P)	Ctrl+P				
終了(X)	Alt+X				
ﾊﾞｰｼﾞｮﾝ情報(A)					

(2) 『ﾒﾆｭｰﾊﾞｰ』→『ﾌｧｲﾙ(F)』→『**名前を付けて保存(A)**』と操作するか Ctrl+A と押下した場合、ComanndButton コントロール(書き込み)を Click した時と同等の動作をする。

(3) 『ﾒﾆｭｰﾊﾞｰ』→『ﾌｧｲﾙ(F)』→『**印刷(P)**』と操作するか Ctrl+P と押下した場合、**メインウィンドウ**のテキストペインを、任意のプリンタに印刷できる。ただし、**応用 6-4** で作成した ComanndButton コントロール(印刷)は削除する。

(4) 『ﾒﾆｭｰﾊﾞｰ』→『ﾌｧｲﾙ(F)』→『**終了(X)**』と操作するか Alt+F4 と押下した場合、ComanndButton コントロール(終了)を Click した時と同等の動作をする。

(5) 『ﾒﾆｭｰﾊﾞｰ』→『**表示(V)**』→『ﾌｫﾝﾄ(F)』と操作するか Ctrl+V と押下した場合、**メインウィンドウ**のテキストペインのフォントを、任意のフォントに変更できる。ただし、**応用 6-3** で作成した ComanndButton コントロール(ﾌｫﾝﾄ)は削除する。

(6) 『ﾒﾆｭｰﾊﾞｰ』→『**ﾊﾞｰｼﾞｮﾝ情報(A)**』と操作するか Alt+A と押下した場合、MsgBox 関数で次のように表示する。

【実装】

(1) テキスト･P506 メニューの作成方法を参考に、frmWriter フォームに **Menu コントロール**を配置する。ただし、今回はコントロール配列を使用しない。

(2) **Menu コントロール**を『ﾒﾆｭｰﾊﾞｰ』→『ﾌｧｲﾙ(F)』→『**名前を付けて保存(A)**』と操作して、コードウィンドウを開き、このメニュー項目に必要な**イベントプロシージャ**を作成する。

(3) **機能**(3)～(6)について、前述(2)を参考に**イベントプロシージャ**を作成する。

【考察】

応用 6-6 応用 5-3 と応用 6-5 を利用して、任意の順編成ファイルを読み込んで画面に表示したり、キーボードから入力した文字列を、任意の順編成ファイルへ書き込むアプリケーションを作成しなさい


D:¥Home¥User1XX¥VB¥Reidai6-6


Project1(TextEditor.vbp)


frmSplash(Splash.frm) … Form1


frmEditor(Editor.frm) … Form2

frmOpenDLG(OpenDLG.frm) … Form3


frmSaveDLG(SaveDLG.frm) … Form4

Module1(Editor.bas) … 標準モジュール

プロジェクトの設定項目

スタートアップの設定	frmSplash … Form1
タイトル	テキストエディター(応用 6-6)
アイコン	CRDFLE13.ICO
Picture のセットアップされている path	
¥¥OkaGp200¥Public¥Microsoft Visual Basic¥GRAPHICS¥ICONS¥OFFICE	

【機能】

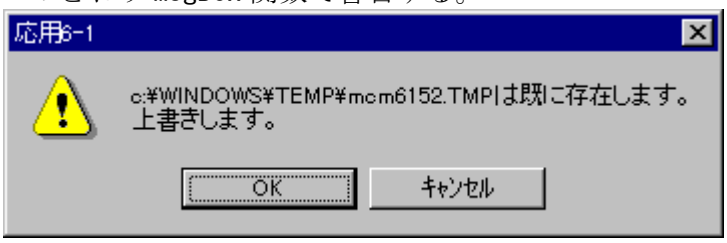
(1) アプリケーションが起動された時、画面中央に**スプラッシュスクリーン**を 5 秒間表示する。
(2) **スプラッシュスクリーン**の表示中は、アプリケーションを**タスクバー**に登録しない。
(3) **スプラッシュスクリーン**は、リサイズしない。
(4) **スプラッシュスクリーン**の非表示後、画面中央に**メインウィンドウ**を表示する。
(5) **メインウィンドウ**の CommandButton コントロール(新規作成)を Click すると、TextBox コントロール(1)、TextBox コントロール(2)及びテキストペインを初期化する。
(6) **メインウィンドウ**の CommandButton コントロール(読み込み)を Click すると、**開くﾀﾞｲｱﾛｸﾞ**(frmOpenDLG)を画面中央に表示する。
(7) **メインウィンドウ**の CommandButton コントロール(書き込み)を Click すると、**保存ﾀﾞｲｱﾛｸﾞ**(frmSaveDLG)を画面中央に表示する。
(8) **メインウィンドウ**の CommandButton コントロール(終了)を Click すると、アプリケーションは終了する。
(9) **メインウィンドウ**の TextBox コントロール(1)に、読み込んだ或いは書き込んだファイルのパスとファイル名を表示する。
(10) 順編成ファイルの読み込み中或いは書き込み中は、**メインウィンドウ**の TextBox コントロール(2)に、"**読み込み中**"或いは"**書き込み中**"と表示する。
(11) 順編成ファイルの読み込み或いは書き込みが完了した時、**メインウィンドウ**の TextBox コントロール(2)に、"**読み込み完了**"或いは"**書き込み完了**"と表示する。
(12) **開くﾀﾞｲｱﾛｸﾞ**と**保存ﾀﾞｲｱﾛｸﾞ**は、**タスクバー**に登録しない。
(13) **開くﾀﾞｲｱﾛｸﾞ**と**保存ﾀﾞｲｱﾛｸﾞ**は、リサイズしない。

(14) **開くﾀﾞｲｱﾛｸﾞ**と**保存ﾀﾞｲｱﾛｸﾞ**の機能は、**応用 5-2･機能**(1)～(5)の機能を持つ。

(15) **開くﾀﾞｲｱﾛｸﾞ**と**保存ﾀﾞｲｱﾛｸﾞ**の CommandButton コントロール(キャンセル)を Click すると、ファイル選択処理を中止し、**メインウィンドウ**に戻る。

(16) **開くﾀﾞｲｱﾛｸﾞ**と**保存ﾀﾞｲｱﾛｸﾞ**では、DriveListBox コントロールが操作された時の、実行時のエラー処理を行う。

(17) **保存ﾀﾞｲｱﾛｸﾞ**で、保存するファイルを決定する時、既にそのファイルが存在する場合は、次のとおり MsgBox 関数で警告する。



(18) **メインウィンドウ**のファイルを操作する時の、実行時のエラー処理を行う。

(19) ☒を Click してアプリケーションを終了させても、ファイルを正しくクローズする。

(20) **メインウィンドウ**をリサイズ操作した場合、各コントロールの大きさや配置位置を調整する。

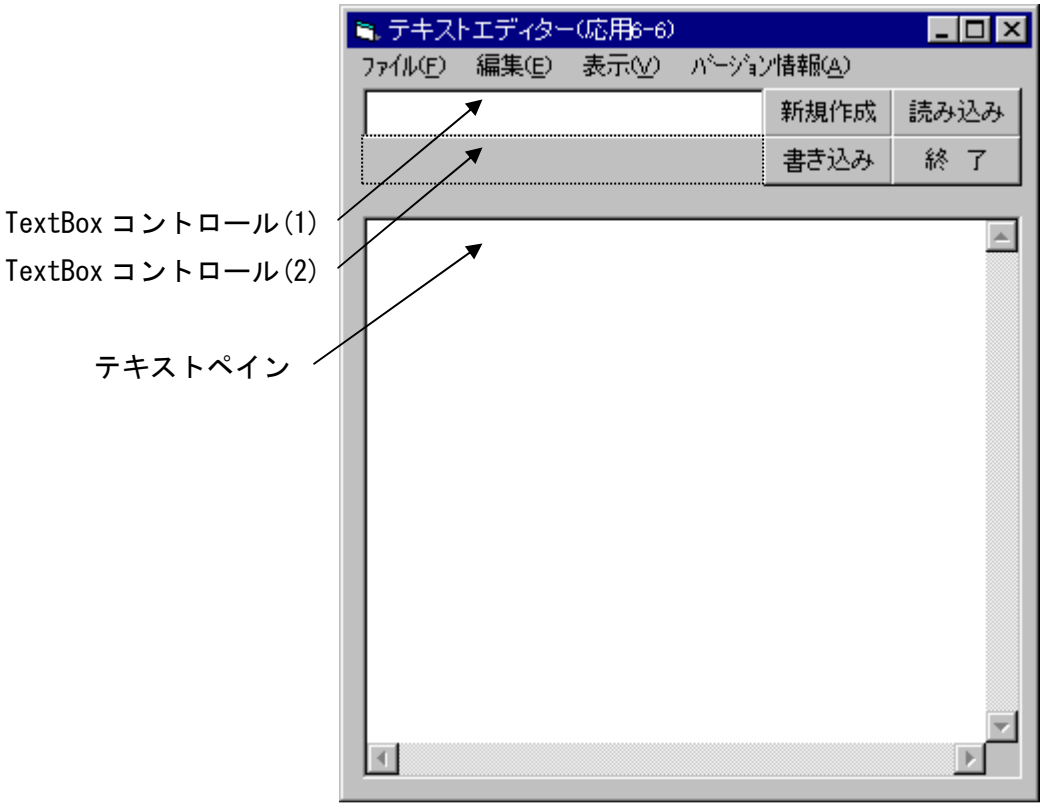
(21) **メインウィンドウ**のテキストペインのフォントを、任意のフォントに変更できる。

(22) **メインウィンドウ**のテキストペインを、任意のプリンタに印刷できる。

(23) **メインウィンドウ**の **Menu コントロール**を追加し、メニューからも操作できる。

Menu コントロールの各メニュー項目は、**応用 6-5** のとおりである。

〔アプリケーションの外観〕



応用 6-7 応用 6-6 に、次のとおりに機能を追加しなさい。

【機能】

(1) **メインウィンドウ**の **Menu コントロール**から、テキストペイン上の文字列の**切り取り**(Cut)、**コピー**(Copy)及び**貼り付け**(Paste)の操作ができる。

(2) テキストペイン上で切り取り及びコピーした文字列を、他のアプリケーションに貼り付けることができる。

(3) 他のアプリケーション上で切り取り及びコピーした文字列を、本アプリケーションに貼り付けることができる。

【解説】

Microsoft Windows 上で動作するアプリケーション同士でテキストやグラフィックスの切り取り、コピー及び貼り付け機能を利用する場合、**Clipboard オブジェクト**を使用する。

Clipboard オブジェクトは、すべての Windows 用アプリケーションによって共有される領域**(clipboard)**に、テキストやグラフィックスのデータを保持する。また、この **Clipboard オブジェクト**上には、データの形式が異なる場合は複数のデータを保持することができるが、同じ形式のデータの場合は上書きされ元のデータは消失する。

Clipboard オブジェクトのメソッド

Clear	Clipboard の内容を初期化(クリア)する。
GetData	Clipboard 上に保持されているピクチャーを返す。
GetFormat	Clipboard に指定した形式と一致するアイテムがあるかどうかを示す整数値を返す。
GetText	Clipboard 上に保持されている文字列を返す。
SetData	Clipboard にピクチャーを挿入する。
SetText	Clipboard に文字列を挿入する。

【構文】

***object*.Clear**

object　　… 対象となるオブジェクトのオブジェクト式を指定。

解説：ListBox コントロール、ComboBox コントロール及び **Clipboard** の内容を初期化(クリア)する。

【Memo】

【構文】

object*.SetText *data, format

object　　… 対象となるオブジェクトのオブジェクト式を指定。

data　　… **Clipboard オブジェクト**に挿入する文字列データを指定。

format　　… **Clipboard オブジェクト**の形式を示す定数又は値を指定。

format の設定値

vbCFLink	&HBF00	DDE(Dynamic Data Exchange)通信の情報
vbCFRTF	&HBF01	リッチテキスト形式
vbCFText	1	テキスト形式(ﾃﾞｨﾌｫﾙﾄ値)

解説：指定した **Clipboard オブジェクト**の形式で、**Clipboard オブジェクト**に文字列データを挿入する。

【Memo】

【構文】

***object*.GetText(*format*)**

object　… 対象となるオブジェクトのオブジェクト式を指定。

format　… **Clipboard オブジェクト**の形式を示す定数又は値を指定。設定値は、**SetText** を参照。

解説：**Clipboard オブジェクト**から、指定した **Clipboard オブジェクト**の形式で文字列を返す。

【Memo】

TextBox コントロールの Text プロパティ内の文字列について、挿入ポインタ、挿入範囲及び選択した文字列の設定・取得の操作は、SelStart プロパティ、SelLength プロパティ及び SelText プロパティを使用する。ただし、これらのプロパティは実行時のみ操作可能である。

【構文】

***object*.SelStart [= *index*]**

object　… オブジェクトへの参照を表すオブジェクト式を指定。

index　… 選択されている文字列の開始点を示す数式を指定。0～テキストの全長。

解説：選択された文字列の開始点を設定・取得する。文字列が選択されていない時は、文字列を挿入する位置を示す。

【Memo】

【構文】

***object*.SelLength [= *number*]**

object　… オブジェクトへの参照を表すオブジェクト式を指定。

number　… 選択されている文字列の長さを表す数式を指定。0～テキストの全長。

解説：選択された文字列の長さを設定・取得する。

【Memo】

【構文】

***object*.SelText [= *value*]**

object　… オブジェクトへの参照を表すオブジェクト式を指定。

value　… 選択されている文字列を含む文字列式を指定。

解説：現在選択されている文字列を含む文字列を設定・取得する。文字が選択されていない場合、長さ 0 の文字列""を返す。

【Memo】

【実装】

(1) **Menu コントロール**を『ﾒﾆｭｰﾊﾞｰ』→『**編集(E)**』→『**切り取り(T)**』と操作して、コードウィンドウを開き、このメニュー項目に必要なイベントプロシージャを作成する。
① **Clipboard** の内容を初期化(クリア)する。
② **Clipboard** に、TextBox コントロール上の選択した文字列を挿入する。
③ TextBox コントロール上の選択した文字列を削除する。

(2) **Menu コントロール**を『ﾒﾆｭｰﾊﾞｰ』→『**編集(E)**』→『ｺﾋﾟｰ(C)』と操作して、コードウィンドウを開き、このメニュー項目に必要なイベントプロシージャを作成する。
① **Clipboard** の内容を初期化(クリア)する。
② **Clipboard** に、TextBox コントロール上の選択した文字列を挿入する。

(3) **Menu コントロール**を『ﾒﾆｭｰﾊﾞｰ』→『**編集(E)**』→『**貼り付け(P)**』と操作して、コードウィンドウを開き、このメニュー項目に必要なイベントプロシージャを作成する。
① **Clipboard** 上の文字列を取得し、TextBox コントロール上の選択した文字列に設定する。

【Memo】

【追加機能】。

(1) **メインウィンドウ**の **Menu コントロール**に、次のメニュー項目を追加する。

編集(E)	
切り取り(T)	Ctrl+X
ｺﾋﾟｰ(C)	Ctrl+C
貼り付け(P)	Ctrl+V
検索(F)	Ctrl+K
置換(E)	Ctrl+H

編集(E)～**貼り付け(P)**までは、**応用 6-5** の仕様のとおり。

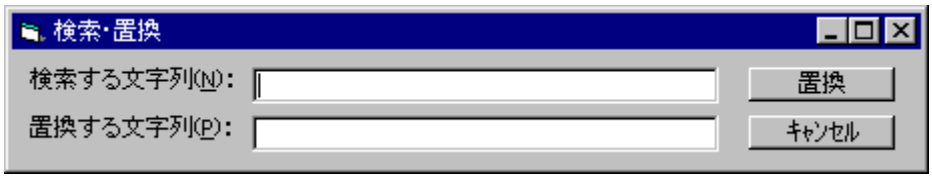
検索(F)と**置換(E)**を追加する。

(2) 『ﾒﾆｭｰﾊﾞｰ』→『**編集(E)**』→『**検索(F)**』と操作するか Ctrl+K と押下した場合、**メインウィンドウ**のテキストペイン内の文字列検索を行う。検索する文字列は、InputBox 関数により取得する。

(3) **メインウィンドウ**のテキストペイン内の文字列検索の結果、文字列が存在した場合は、該当の文字列を反転表示(選択した状態)する。

(4) **メインウィンドウ**のテキストペイン内の文字列検索の結果、文字列が存在しなかった場合は、MsgBox 関数で次のとおりに表示する。

(5) 『ﾒﾆｭｰﾊﾞｰ』→『**編集**(E)』→『**置換**(H)』と操作するか Ctrl +H と押下した場合、**メインウィンドウ**のテキストペイン内の文字列検索・置換を行う。検索する文字列と置換する文字列は次のフォームから取得する。



オブジェクト名　… frmSearch

(6) **メインウィンドウ**のテキストペイン内の文字列検索・置換の結果、文字列が存在した場合は、該当の文字列を置換する文字列を置換え反転表示(選択した状態)する。

(7) **メインウィンドウ**のテキストペイン内の文字列検索・置換の結果、文字列が存在しなかった場合は、MsgBox 関数で前述(4)のとおりに表示する。

【実装】

(1) **Menu コントロール**を『ﾒﾆｭｰﾊﾞｰ』→『**編集**(E)』→『**検索**(F)』と操作して、コードウィンドウを開き、このメニュー項目に必要なイベントプロシージャを作成する。

① 文字列の検索には、Instr 関数を使用する。

【構文】

InStr([*start*,]*string1*, *string2*[, *compare*])

start　… 検索の開始位置を表す数式を指定。省略すると、先頭の文字から検索される。

string1　… 検索対象となる文字列式を指定。

string2　… 引数 *string1* 内を検索する文字列式を指定。

compare　… 文字列比較の比較モードを指定する番号を設定。

戻り値　… 最初に見つかった文字位置を示すバリアント型の値。

解説：*string1* の中から指定した *string2* を検索し、最初に見つかった文字位置(先頭からその位置までの文字数)を返す文字列処理関数。

【Memo】

② 検索する文字列が存在した場合、**SelStart プロパティ**と **SelLength プロパティ**の操作により、該当の文字列を反転表示(選択した状態)する。

(2) **Menu コントロール**を『ﾒﾆｭｰﾊﾞｰ』→『**編集(E)**』→『**置換(E)**』と操作して、コードウィンドウを開き、このメニュー項目に必要なイベントプロシージャを作成する。

① Instr 関数によって、置換される文字列を検索する。

② 置換される文字列が存在した場合、**SelStart プロパティ**と **SelLength プロパティ**の操作により、該当の文字列を反転表示(選択した状態)する。

③ **SelText プロパティ**の操作により、該当の文字列を置換する。

【課題】

このアプリケーションを完成させ、実行形式のプログラムを所定の期日までに所定のフォルダに提出しなさい。

【考察】

補足１ 次のとおりセットアッププログラムを作成しなさい。

【解説】

Visual Basic で作成したアプリケーション(実行形式ファイル)を、アプリケーションを開発したコンピュータから、別のコンピュータにコピーしても動作しない場合がある。

これはアプリケーションの実行に必要な依存ファイルなどが、コピー先のコンピュータにセットアップされていないからである。

この依存ファイルは、**VB ランタイムライブラリ**、**カスタムコントロール**、**Active X コントロール**、**OLE コントロール**及び**Windows コントロール**などの複数のファイルで構成されている。

また、これらの依存ファイルは、作成したアプリケーションにより異なるなどの点から、手作業でコピーなどの操作を行うことは煩雑であり困難となる場合がある。

Visual Basic では、作成したアプリケーション、実行に必要な依存ファイルやデータファイルなどをまとめて、別のコンピュータにセットアップしてくれるプログラムを作成する**セットアップウィザード**が用意されている。

Visual Basic **セットアップウィザード**によるセットアッププログラムの作成手順を次に示す。

(1) [ﾌﾟﾛｼﾞｪｸﾄﾌﾟﾛﾊﾟﾃｨ]ダイアログボックスで、**ﾊﾞｰｼﾞｮﾝ番号**や**ﾊﾞｰｼﾞｮﾝ情報**などのアプリケーション情報を設定する。

(2) プログラムをコンパイルして、アプリケーション(実行形式ファイル)を作成する。

(3) Visual Basic **セットアップウィザード**で、セットアッププログラムを作成する。

【手順】

(1) セｯﾄｱｯﾌﾟｳｨｻﾞｰﾄﾞを起動する。

(2) [**ﾌﾟﾛｼﾞｪｸﾄとｵﾌﾟｼｮﾝの選択**]ダイアログボックスでは、配布するアプリケーションのプロジェクトファイルを設定し、セットアッププログラムのオプションを選択する。

(例)

“D:¥Home¥User1XXX¥VB¥Reidai6¥TextEditor.vbp” … **ｱﾌﾟﾘｹｰｼｮﾝのﾌﾟﾛｼﾞｪｸﾄﾌｧｲﾙ**を設定

◉**ｾｯﾄｱｯﾌﾟﾌﾟﾛｸﾞﾗﾑを作成(S)** … **ｾｯﾄｱｯﾌﾟﾌﾟﾛｸﾞﾗﾑを作成**を設定

☑**依存ﾌｧｲﾙを生成(G)** … **依存ﾌｧｲﾙを生成**を設定

【Memo】

.....

.....

(3) [**配布方法**]ダイアログボックスでは、セットアッププログラムを保存するメディアを選択する。

(例)

◉**ﾌﾛｯﾋﾟｰﾃﾞｨｽｸ(L)** … 複数枚になる可能性がある。

【Memo】

.....

.....

(4) [**ﾌﾛｯﾋﾟｰﾃﾞｨｽｸ**]ダイアログボックスでは、ドライブレターと記録容量を選択する。

(例)

ﾌﾛｯﾋﾟｰﾄﾞﾗｲﾌﾞ(L): A: … ﾄﾞﾗｲﾌﾞA:を設定

ﾃﾞｨｽｸｻｲｽﾞ(S): 1.44MB … MF-2HD タイプを設定

【Memo】

(5) [**依存情報の確認**]ダイアログボックスでは、Microsoft 社の配布権限に関するライセンス契約書を確認の上、実際に配布するファイルに☑を付ける。

(例)

☑C:¥WINNT¥SYSTE32¥COMDLG32.OCX … **コモンダイアログコントロール**

【Memo】

(6) [ｾｯﾄｱｯﾌﾟｳｨｻﾞｰﾄﾞ]の各ダイアログボックスに対して、内容を確認して[OK]又は[次へ(N)>](或いは[完了(F)])を Click する。

【Memo】

【課題】

- TextEditor セットアッププログラムを作成し、別のコンピュータにセットアップしなさい。セットアップ完了後、アプリケーションの動作確認をしなさい。

【考察】

例題 7 Microsoft Excel ワークシートを表示するアプリケーションを作成しなさい。


D:¥Home¥User1XX¥VB¥Reidai7

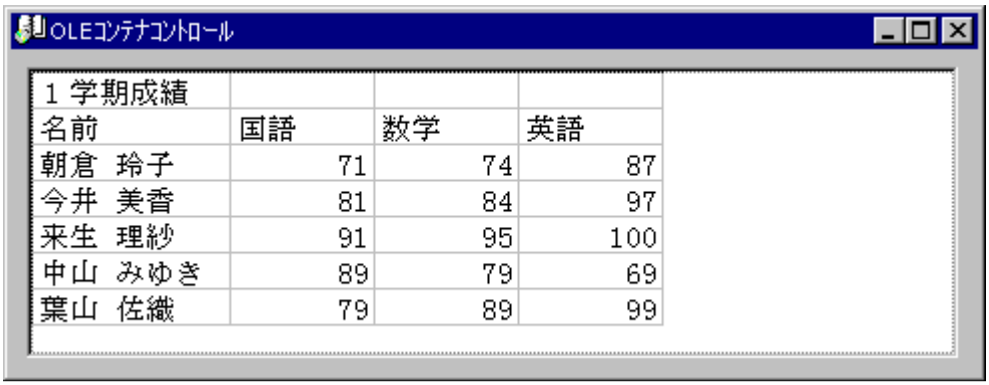

Project1(OleExcel.vbp)


Form1(OleExcel.frm)

プロジェクトの設定項目

スタートアップの設定	Form1
タイトル	OLEｺﾝﾃﾅｺﾝﾄﾛｰﾙ
アイコン	BOOK06.ICO
Picture のセットアップされている path	
¥¥OkaGp200¥Public¥Microsoft Visual Basic¥GRAPHICS¥ICONS¥Writing	

〔アプリケーションの外観〕



【解説】

Visual Basic のアプリケーションのフォームに、他のアプリケーションオブジェクト(Excel ワークシート、Word 文書など)を挿入する場合、**OLE(Object Linking and Embedding)**を利用する。

OLE には、このアプリケーションオブジェクトを挿入するプレースホルダーとして、**OLE コンテナコントロール**がある。

OLE コンテナコントロールは挿入オブジェクトを表示できるが、挿入オブジェクトのデータへの参照はできない。

また、**OLE コンテナコントロール**で利用可能なオブジェクトは、そのコンピュータにインストールされているアプリケーション(Excel、Word など)に依存する。

OLE コンテナコントロールのオブジェクトの挿入方法には、**リンクモード(Link mode)**と**エンベッドモード(Embed mode)**がある。

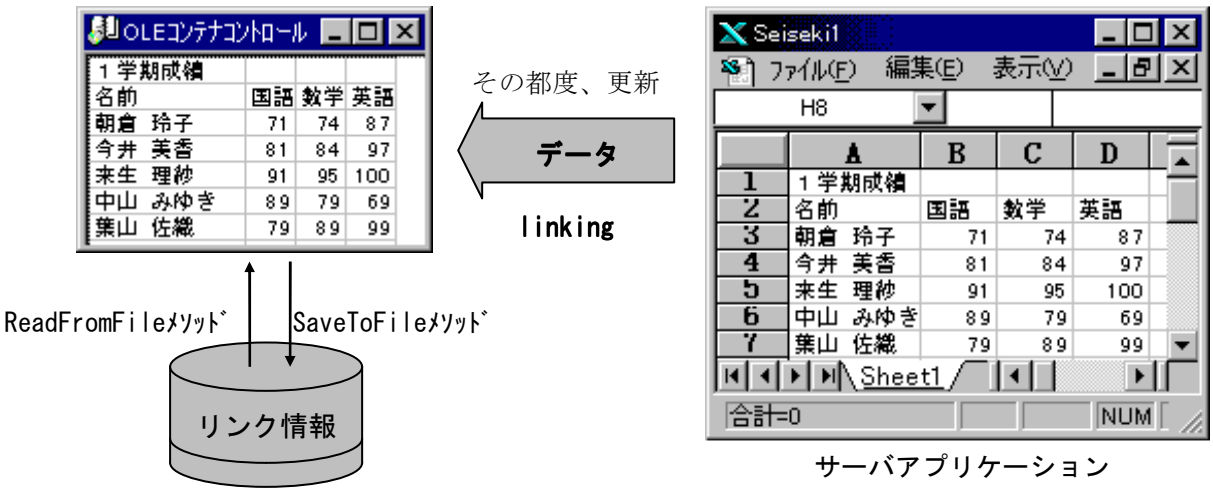
① **リンクモード(Link mode)**

リンクされたオブジェクトのデータ(Excel ワークシート、Word 文書など)は、作成元のアプリケーション(Excel、Word など)に格納・管理される。

OLE コンテナコントロールを保存する場合、**SourceDoc プロパティ**や **SourceItem プロパティ**などのリンク情報とデータのイメージがデータファイルとして保存される。

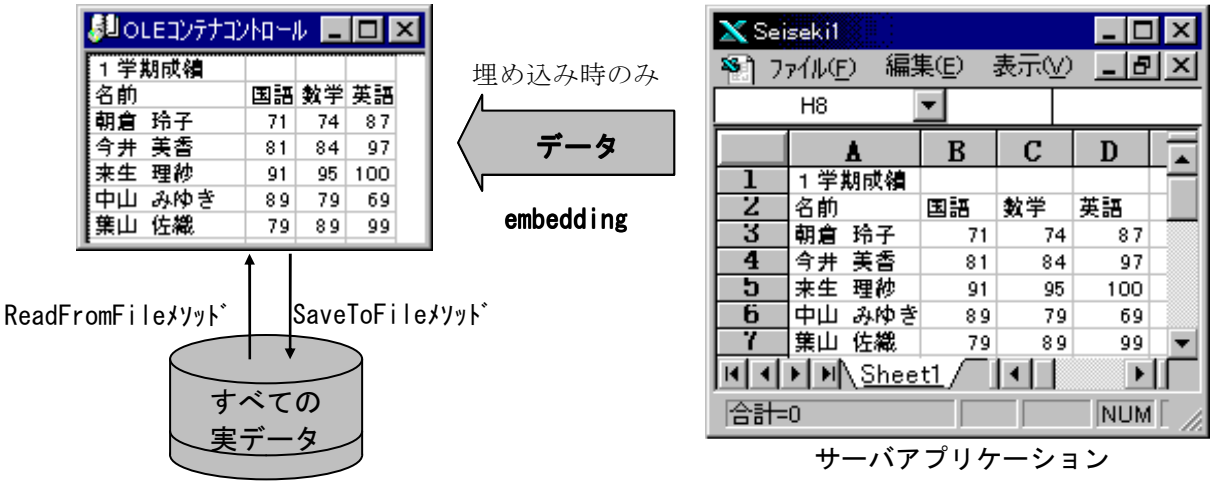
したがって、作成元のアプリケーションでリンクされたオブジェクトのデータを更新すると、その結果が **OLE コンテナコントロール**に反映される。

なお、作成元のアプリケーションを、**サーバアプリケーション**という。



② **エンベッドモード(Embed mode)**

埋め込まれたオブジェクトのすべてのデータは、Visual Basic のアプリケーションにコピーされているため **OLE コンテナコントロール**で格納・管理される。



OLE コンテナコントロールの主なプロパティ

プロパティ	説　明
Class	OLE コンテナコントロールに挿入されるオブジェクトの種類(**クラス名**)を設定。
Container	Form オブジェクト上のコントロールのコンテナを設定。
DataField	**カレントレコード**のフィールドにコントロールを連結する値(**フィールド名**)を設定。
DataSource	データを結合する**レコードセット**を設定。
DisplayType	オブジェクトを表示するか、**アイコン**として表示するかを設定。
NegotiateMenus	フォーム上のオブジェクトのメニューを、フォームのメニューに取り込むかを設定。
OLEType	**エンベッドモード**か、**リンクモード**かを取得。
OLETypeAllowed	OLE コンテナに挿入するオブジェクトの、挿入方法を設定。
OLEDropAllowed	ドラッグドロップのターゲットにするかを設定。
SizeMode	**挿入オブジェクト**のアイコン又はデータの表示形式を設定。
SourceDoc	**リンクオブジェクト**の作成時に、リンクするソースファイルを設定。
SourceItem	ファイルの一部をリンクする時に、リンクする**データ項目**を設定。

OLE リンクオブジェクトの作成方法には、「**デザイン時に作成**」する方法と、「**実行時に作成**」する方法がある。実行時に、**OLE リンクオブジェクト**を作成するには、次の**メソッド**を使用する。

OLE コンテナコントロールの主な**メソッド**

メソッド	説　明
CreateEmbed	**埋め込みオブジェクト**を作成。
CreateLink	**リンクオブジェクト**を作成。
DoVerb	オブジェクトに対し、指定の OLE 動作を実行。
InsertObjDlg	「**オブジェクトの挿入**」ダイアログボックスを表示して、オブジェクトを挿入。
PasteSpecialDlg	「**形式を選択して貼り付け**」ダイアログボックスを表示して、**クリップボード**のデータをコピー。
ReadFormFile	ファイルからデータをオブジェクトに読み込む。
SaveToFile	ファイルにオブジェクトデータを保存。

【実装】

(1) **ツールボックス**上の OLE  を Click して、Form1 フォームに配置する。

(2) サブテキスト･P42 図 1-73 を参考に、「**ｵﾌﾞｼﾞｪｸﾄの挿入**」ダイアログボックスの「**ﾌｧｲﾙから**(F)」OptionButton を check し、[参照(B)...] を Click する。

【Memo】

(3) 「**参照**」ダイアログボックスで、次のファイルを選択し、[開く(O)] を Click する。

"¥¥OkaGp200¥Public¥Microsoft Visual Basic¥Example¥Seiseki1.XLS"

(4) 「**ﾌｧｲﾙ**(E):」のファイル名と、「**ﾘﾝｸ**(L):」を確認して、[OK] を Click する。

【Memo】

【課題】

(1) アプリケーションの起動後、**OLE コンテナコントロール**を DoubleClick しなさい。

【Memo】

(2) サブテキスト･P43 を参考に、Form1 フォームの **NegotiateMenus プロパティ**を False に変更して、前述(1)の操作をしなさい。

【Memo】

(3) CommondPrompt を起動して、TYPE コマンドと VIEW.EXE により OleExcel.frx の内容を確認しなさい。

【Memo】

(4) **サーバアプリケーション**側でデータを変更して上書き保存し、本アプリケーションを再起動しなさい。

【Memo】

(5) **実装**(4)・「□ﾘﾝｸ(L):」を「☑ﾘﾝｸ(L):」に変更して、本アプリケーションを再起動しなさい。

【手順】

① Form1 上の **OLE コンテナコントロール**を、Delete で削除する。

② 再度、**ツールボックス**上の OLE を Click して、Form1 フォームに配置する。

③ **実装**(2)及び(3)の操作を行う。

④ 「ﾌｧｲﾙ(E):」のファイル名と、、「☑ﾘﾝｸ(L):」を確認して、 OK を Click する。

【Memo】

⑤ 本アプリケーションを起動する。

⑥ Microsoft Excel を起動し、次のファイルを開く。
"¥¥OkaGp200¥Public¥Microsoft Visual Basic¥Example¥Seiseki1.XLS"

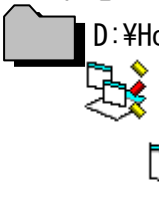
⑦ Microsoft Excel のセルの内容を変更する。

⑧ 本アプリケーションの表示内容を確認する。

【Memo】

【考察】

例題 7-1 任意のオブジェクトを表示するアプリケーションを作成しなさい。



プロジェクトの設定項目

スタートアップの設定	Form1
タイトル	OLEﾘﾝｸｵﾌﾞｼﾞｪｸﾄ
アイコン	OPENFOLD.ICO
Picture のセットアップされている path	
¥¥OkaGp200¥Public¥Microsoft Visual Basic¥GRAPHICS¥ICONS¥Win95	

〔アプリケーションの外観〕

【解説】

例題 7 は、**OLEリンクオブジェクト**を「**デザイン時に作成**」する方法でアプリケーションを作成した。今回は、**OLEリンクオブジェクト**を「**実行時に作成**」する方法でアプリケーションを作成する。

【実装】

(1) **ツールボックス**上の OLE を Click して、Form1 フォームに配置する。

(2) 「**ｵﾌﾞｼﾞｪｸﾄの挿入**」ダイアログボックスが表示されたら、直ちに [ｷｬﾝｾﾙ] を Click し、オブジェクトの挿入を中止する。 **Point:**「**実行時に作成**」する場合は必須の操作。

【Memo】

.....

.....

(3) テキスト･P506 メニューの作成方法を参考に、Form1 フォームに Menu コントロールを配置する。

(4) Menu コントロールを『**ﾒﾆｭｰﾊﾞｰ**』→『**ｵﾌﾞｼﾞｪｸﾄの挿入**』と操作して、コードウィンドウを開き、このメニュー項目に必要なイベントプロシージャを作成する。

(例)

```
Option Explicit
Private Sub mnuLinkOLE_Click()
'「ｵﾌﾞｼﾞｪｸﾄの挿入」ダイアログボックスでｵﾌﾞｼﾞｪｸﾄを選択して表示
    OLE1.InsertObjDlg       '「ｵﾌﾞｼﾞｪｸﾄの挿入」ダイアログボックス表示
    OLE1.DoVerb             ' OLE 動作の実行
End Sub
```

【構文】

***object*.InsertObjDlg**

object　　… 対象となるオブジェクトのオブジェクト式を指定。

解説：「**ｵﾌﾞｼﾞｪｸﾄの挿入**」ダイアログボックスを表示する。実行時に、ユーザが**サーバアプリケーション**と形式(**リンクモード**又は**埋め込みモード**)を選択して、**リンクオブジェクト**又は**埋め込みオブジェクト**を作成できる。このダイアログボックスを使う場合、**OLETypeAllowed プロパティ**で作成できるオブジェクトの形式を指定する。また、新しいオブジェクトを作成する時は、**クラス名**に関連する**サーバアプリケーション**が、オペレーティングシステムに正しく登録されている必要がある。

【構文】

***object*.DoVerb ([*verb*])**

object　　… 対象となるオブジェクトのオブジェクト式を指定。

verb　　… **OLEコンテナコントロール**のオブジェクトが実行する**OLE動詞**を指定。この引数の値は、すべてのオブジェクトがサポートする標準の OLE 動詞のいずれか、又は、**ObjectVerbs プロパティ配列**のインデックスである。

※ **OLE動詞**:オブジェクトに対して実行できるアクション。

解説：**AutoActivate プロパティ**に 2(Double Click)を設定すると、ユーザーがコントロールを Double Click した時に、自動的に現在のオブジェクトがアクティブになる。

標準的な **OLE 動詞**の値

定数	値	内　容
vbOLEPrimary	0	オブジェクトの既定の動作をする。
vbOLEShow	-1	編集のためにオブジェクトをアクティブにする。
vbOLEOpen	-2	別のアプリケーションウィンドウでオブジェクトを開く。
vbOLEHide	-3	**埋め込みオブジェクト**として表示。
vbOLEUIActivate	-4	オブジェクトが埋め込み先編集をサポートしている場合、埋め込み先編集のためにオブジェクトをアクティブにして、ユーザーインターフェイスツールを表示する。
vbOLEInPlaceActivate	-5	ユーザーが OLE コンテナコントロールにフォーカスを移動すると、オブジェクトのウィンドウが作成され、オブジェクトを編集できるように準備する。
vbOLEDiscardUndoState	-6	オブジェクトをアクティブにしたまま、オブジェクトのアプリケーションが操作を取り消すことができる変更のすべてを破棄するときに使用する。

※ 詳細は、Visual Basic のヘルプを参照。

【課題】

(1) **OLE 動詞**に、**vbOLEOpen** を指定して動作させなさい。

【Memo】

(2) OLE 動詞に、他の定数を指定して動作させなさい。

【Memo】

例題 7-2 次の機能の説明を読んで、Excel ワークシートを表示するアプリケーションを作成しなさい。

D:¥Home¥User1XX¥VB¥Reidai7

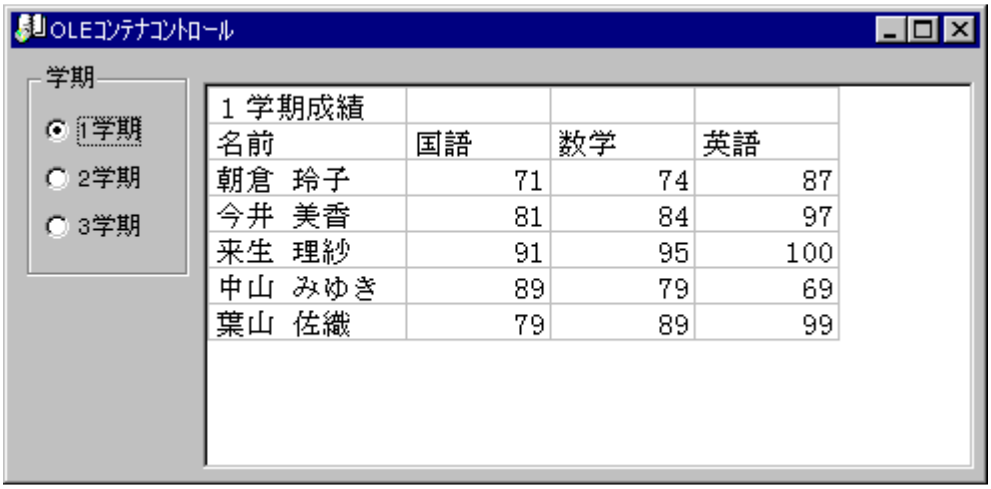
Project1(OleExcel1.vbp)

Form1(OleExcel1.frm)

プロジェクトの設定項目

スタートアップの設定	Form1
アイコン	BOOK06.ICO
Picture のセットアップされている path	
¥¥OkaGp200¥Public¥Microsoft Visual Basic¥GRAPHICS¥ICONS¥Writing	

〔アプリケーションの外観〕



【機能】

(1) 1 学期～3 学期の成績表が、Seiseki1.xls～Seiseki3.xls として Excel ファイルで保存されている。この Excel ファイルを、画面を切り替えて表示できる。

(2) OptionButton コントロールを操作すると、画面を切り替えることができる。

サンプルアプリケーションが保存されている path とアプリケーション名

¥¥XXXXXXXX¥Public¥Microsoft Visual Basic¥Example¥OLE_Rei1.EXE

【実装】

(1) **ツールボックス**上の OLE を Click して、Form1 フォームに配置する。

(2) 「**ｵﾌﾞｼﾞｪｸﾄの挿入**」ダイアログボックスが表示されたら、直ちに キャンセル を Click し、オブジェクトの挿入を中止する。 **Point:「実行時に作成」**する場合は必須の操作。

【Memo】

.....

.....

(3) テキスト･P340 図 5-10 を参考に、Form1 フォームに **Frame コントロール**を配置する。

(4) テキスト･P343 を参考に、**Frame コントロール**上に OptionButton コントロールを配置する。ただし、1 学期～3 学期を**コントロール配列**として作成する。

【解説】

コントロール配列とは、配列の概念をコントロールに反映させたものである。すなわち、同じ種類の複数のコントロールに対して、同じ**オブジェクト名**(Name プロパティ)を重複して設定し、個々の識別は**要素番号(Index プロパティ)**で行う。

コントロール配列を使用するメリットは、次のとおりである。

① イベントプロシージャを共通化することができる。

② プログラムの中で、**コントロール配列**の要素を追加・削除(Load/Unload)できる。

コントロール配列の作成の手順は、次のとおりである(OptionButton コントロールの例)。

① OptionButton コントロールを、オブジェクト上に配置する。

② 前述①の OptionButton コントロールをポイントして右 Click し、[ｺﾋﾟｰ(C)]を選択する。

③ オブジェクトをポイントして右 Click し、[**貼り付け**(P)]を選択する。

④ テキスト･P396 図 5-56 を参考に、[はい(Y)] を Click する。

【Memo】

(5) Form_Load()イベントプロシージャでは、次の処理を行う。

① **Class プロパティ**に、**挿入オブジェクト**の種類を設定する。

② Seiseki1.xls ファイルの内容から、**OLE リンクオブジェクト**を作成する。

③ OptionButton コントロール配列(1 学期)を、選択された状態にする。

【構文】

***object*.Class [= *string*]**

object　　… 対象となるオブジェクトのオブジェクト式を指定。

string　　… **挿入オブジェクト**の種類を指定する文字列式を指定。

挿入オブジェクトの種類を指定する文字列式

クラス名	挿入オブジェクトの種類
Excel.Sheet.5	Microsoft Excel ワークシート
Excel.Chart.5	Microsoft Excel グラフ
MSGraph.Chart.5	Microsoft Graph5.0
PowerPoint.Slide.7	Microsoft PowerPoint スライド
Word.Picture.6	Microsoft Word 図
Word.Document.6	Microsoft Word 文書
MA_ClipArt_Gallery.2	Microsoft クリップアート 2.0
SoundRec	WAVE サウンド
Paint.Picture	ビットマップイメージ
WordPad.Document.1	ワードパッド文書

解説：**挿入オブジェクト**の**クラス名**の設定・取得を行う。**クラス名**は、オブジェクトの種

類を定義する。

【構文】

application.Objecttype.version

Objecttype.version

application　… オブジェクトを提供するアプリケーションの名前を指定。

Objecttype　… オブジェクトライブラリで定義されたオブジェクトの名前を指定。

version　… オブジェクト又はオブジェクトを提供するアプリケーションのバージョン番号を指定。

【構文】

object*.CreateLink *sourcedoc, sourceitem

object　… 対象となるオブジェクトのオブジェクト式を指定。

sourcedoc　… オブジェクトを格納しているファイルを指定。

sourceitem　… **リンクオブジェクト**にリンクしているファイルのデータを指定。

解説：指定したファイルの内容から**リンクオブジェクト**を作成し、**OLE コンテナコントロール**に指定したファイルのイメージが表示される。

(例)

```
Private Sub Form_Load()
    OLE1.Class = "Excel.Sheet.5"
    OLE1.CreateLink "C:¥My Documents¥Seiseki1.xls"
    optTerm(0).Value = True
End Sub
```

【Memo】

(6) OptionButton コントロール配列が Click された時の、イベントプロシージャを作成する。

① 選択された OptionButton コントロール配列に合わせて、Excel ファイル(Seiseki1.xls～Seiseki3.xls)の内容から、**OLE リンクオブジェクト**を作成する。

(例)

```
Private Sub optTerm_Click(Index As Integer)
    OLE1.CreateLink "C:¥My Documents¥Seiseki" & (Index + 1) & ".xls"
End Sub
```

【Memo】

【考察】

応用 7 次の機能の説明を読んで、Excel ワークシートを表示するアプリケーションを作成しなさい。


D:¥Home¥User1XX¥VB¥Reidai7


Project1(OleExcel2.vbp)


Form1(OleExcel2.frm)

プロジェクトの設定項目

スタートアップの設定	Form1
アイコン	BOOK06.ICO
Picture のセットアップされている path	
¥¥OkaGp200¥Public¥Microsoft Visual Basic¥GRAPHICS¥ICONS¥Writing	

【機能】

(1) **例題 7-2** の機能(1)と(2)の機能を持つ。

(2) 最初に**リンクオブジェクト**を作成する時、「ﾌｧｲﾙを開く」ダイアログボックスを表示して、ファイルを選択できる。

(3) 「ﾌｧｲﾙを開く」ダイアログボックスで キャンセル が Click された時は、処理を中止しアプリケーションは終了する。

(4) 本アプリケーションが取り扱う Excel ファイルの名前は、次のとおりとする。
"任意の文字列" & 1～3 の数字 1 桁 & ".xls"

サンプルアプリケーションが保存されている path とアプリケーション名
¥¥XXXXXXXX¥Public¥Microsoft Visual Basic¥Example¥OLE_Rei2.EXE

【実装】

(1) **例題 7-2** で作成した、フォーム、コントロール及びプロシージャを再利用する。

(2) **ツールボックス**上の CommonDialog  を Click して、Form1 フォームに配置する。
CommonDialog コントロールは、デザイン時には表示されるが、実行時は非表示となる。
また、コントロールのサイズは、自動的にディフォルト値に設定される。

(3) テキスト･P568 を参考に、Form_Load()イベントプロシージャに「ﾌｧｲﾙを開く」ダイアログボックスを表示するステートメントを追加記述する。
① 「ﾌｧｲﾙを開く」ダイアログボックスの キャンセル の押下を検出するために、エラートラッピング処理を開始する。
② **コモンダイアログボックス**の **CancelError プロパティ**に キャンセル の有効化を設定する。
③ 「ﾌｧｲﾙを開く」ダイアログボックスの[ﾌｧｲﾙ名]リストボックスに Excel ファイルだけを表示するために、**Filter プロパティ**に**見出し**と**フィルタ**を設定する。
この見出しは、「ﾌｧｲﾙの種類(T):」に表示される。

見出しと**フィルタ**の設定

見出し	フィルタ	説　明
Microsoft Excel (*.xls)	*.xls	拡張子が"xls"であるファイルだけを表示する。
すべて (*.*)	*.*	すべてのファイルを表示する。

【構文】

object.Filter [= *description1* |*filter1*|*description2* |*filter2*...]

object　… オブジェクトへの参照を表すオブジェクト式を指定。

description … ファイルの種類を表す文字列式を指定。 ⇐ **見出し**

filter　… ファイル名の拡張子を指定する文字列を指定。 ⇐ **フィルタ**

解説：フィルタを使うと、「**ﾌｧｲﾙを開く**」ダイアログボックスの［ﾌｧｲﾙ名］リストボックスに表示されるファイルの種類を設定することができる。また、複数のフィルタを設定することもできる。

④ **FileName プロパティ**から選択した Excel ファイルのパス及びファイル名を取得する。

【構文】

object.FileName [= *pathname*]

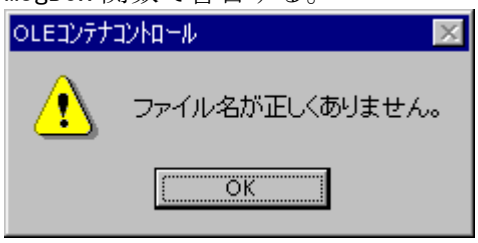
object　… オブジェクトへの参照を表すオブジェクト式を指定。

pathname　… パス及びファイル名を含む文字列式を指定。

ドライブ名:¥フォルダ名¥ファイル名 ⇐ **絶対パス形式**

解説：選択されているファイルのパス及びファイル名を設定・取得することができる。また、有効なネットワークパスとファイル名を指定することもできる。

⑤ 有効な Excel ファイルのパス及びファイル名を取得できなかった場合、次のとおり MsgBox 関数で警告する。



〔⑤ パス及びファイル名が無効〕

〔⑨ ｷｬﾝｾﾙ が Click された場合〕

⑥ **Class プロパティ**に、**挿入オブジェクト**の種類を設定する。

⑦ 選択した Excel ファイルの内容から、**OLE リンクオブジェクト**を作成する。

⑧ OptionButton コントロール配列(1 学期)を、選択された状態にする。

⑨「**ﾌｧｲﾙを開く**」ダイアログボックスで ｷｬﾝｾﾙ が Click された時の**エラートラッピングルーチン**では、上記のとおり MsgBox 関数で警告し、その後 End ステートメントによりアプリケーションを終了させる。

(4) OptionButton コントロール配列が Click された時の、イベントプロシージャを作成する。

【解説】

例題 7-2 では、Excel ファイルのパス及びファイル名を固定(*hard coding*)していたが、今回は任意のファイルを選択できるように設計変更している。

したがって、Excel ファイルのパス及びファイル名に対して次の操作が必要となる。

① **FileName プロパティ**によって取得した Excel ファイルのパス及びファイル名から、1～3 の数字 1 桁と拡張子(*extension*)を除いたパス及びファイル名を取り出す。

② 処理の効率化を考慮すると、上記①の操作は、このイベントプロシージャが最初に実行される 1 回だけで良い。

③ 上記①の操作によって取り出した Excel ファイルのパス及びファイル名、1～3 の数字

1桁(引数 Index から取得)、".(*dot*)"及び拡張子(*extension*)を文字列連結し、これによって示された Excel ファイルの内容から、**OLE リンクオブジェクト**を作成する。

また、上記①～③の操作に必要となるローカル変数の記憶クラスは、**静的(Static)**としておくこと。

※「**ﾌｧｲﾙを開く**」ダイアログボックスのオプション値は、次のとおりである。Flags プロパティに設定し、動作を確認しなさい(「**ﾌｧｲﾙ名を付けて保存**」ダイアログボックスと共通)。

オプション値		説　明
cdlOFNAllowMultiselect	&H200&	複数のファイルを選択可能。
cdlOFNCreatePrompt	&H2000&	ファイルが存在しない時に新規作成するか問合せを行う。
cdlOFNExtensionDifferent	&H400&	取得したファイル名の拡張子が DefaultExt プロパティの拡張子と異なる事を示す。
cdlOFNFileMustExist	&H1000&	既存のファイル名だけを入力できるように設定。
cdlOFNHideReadOnly	&H4&	[**読み取り専用**]チェックボックスを表示しない。
cdlOFNNoChangeDir	&H8&	ダイアログボックスを開いた時、現在のディレクトリを表示。
cdlOFNNoReadOnlyReturn	&H8000&	読み取り専用属性を持たず、読み取り専用ディレクトリにないファイルを取得。
cdlOFNNoValidate	&H100&	無効な文字を含むファイル名を取得。
cdlOFNOverwritePrompt	&H2&	ファイルを上書きするか確認メッセージを表示。
cdlOFNPathMustExist	&H800&	有効なパスだけを入力できるように設定。
cdlOFNReadOnly	&H1&	[**読み取り専用**]チェックボックスを有効にする。
cdlOFNShareAware	&H4000&	共有違反エラーを無視する。
cdlOFNHelpButton	&H10&	ヘルプボタンを表示。
cdlOFNExplorer	&H80000&	エクスプローラ風ダイアログボックスを表示。
cdlOFNNoDereferenceLinks	&H100000&	ショートカットを実行しない。
cdlOFNLongNames	&H200000&	長いファイル名を使用。

オプション値を複数設定する場合は、**Or 演算子**を使用する。

【構文】

***object*.Flags [= *value*]**

object　… オブジェクトへの参照を表すオブジェクト式を指定

value　… [**ﾌｧｲﾙを開く**]、[**ﾌｧｲﾙ名を付けて保存**]の各ダイアログボックスの状態を設定する定数または数値を指定(上記を参照)。

【構文】

***object*.FilterIndex [= *number*]**

object　… オブジェクトへの参照を表すオブジェクト式を指定

number　… フィルタの既定値を示す数式を指定。

解説：二つ以上のフィルタが設定されている場合、フィルタの既定値を設定・取得する。

【構文】

***object*.FileTitle**

object　… オブジェクトへの参照を表すオブジェクト式を指定

解説：パスを含まないファイル名を取得する。

【考察】

【追加機能】

・ **メインウィンドウ**をリサイズ操作した場合、**OLE コンテナコントロール**の大きさを調整する。

【実装】

・ **Form_Resize イベントプロシージャ**に、**OLE コンテナコントロール**の大きさを調整するプログラムコードを記述する(サブテキスト･P10 を参照)。

① フォームのサイズの最小値を確認し、設計したフォームの高さや幅よりも小さくならないこと。

② リサイズ操作に応じて、**OLE コンテナコントロール**の大きさを調整する。なお、その他のコントロールについては、大きさや配置位置の調整をしない。

【考察】

応用 7-1 次の機能の説明を読んで、任意のオブジェクトを表示するアプリケーションを作成しなさい。

D:¥Home¥User1XX¥VB¥Reidai7

Project1(OleLink2.vbp)

Form1(OleLink2.frm)

プロジェクトの設定項目

スタートアップの設定	Form1
アイコン	OPENFOLD.ICO
Picture のセットアップされている path	
¥¥OkaGp200¥Public¥Microsoft Visual Basic¥GRAPHICS¥ICONS¥Win95	

【機能】

(1) Windows で相互利用可能な任意のアプリケーションオブジェクトを、**OLE** によって Visual Basic のフォームに表示する。

(2) 『ﾒﾆｭｰﾊﾞｰ』→『ﾌｧｲﾙ(F)』→『開く(O)』と操作するか Ctrl +O と押下した場合、「ﾌｧｲﾙを開く」ダイアログボックスを表示して、**OLE リンクオブジェクト**のデータファイルから**リンク**

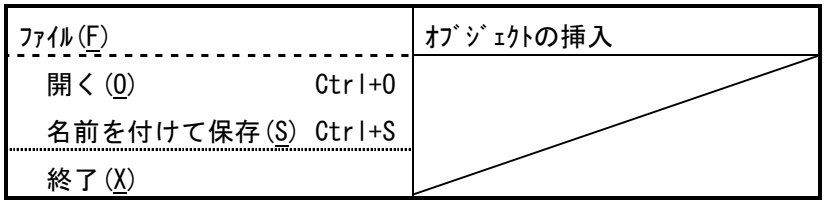
情報などを読み込む。

(3) 『ﾒﾆｭｰﾊﾞｰ』→『ﾌｧｲﾙ(F)』→『**名前を付けて保存**(S)』と操作するか Ctrl +S と押下した場合、**「ﾌｧｲﾙ名を付けて保存」ダイアログボックス**を表示して、**OLE リンクオブジェクト**の**リンク情報**などをデータファイルへ書き込む。

(4) 『ﾒﾆｭｰﾊﾞｰ』→『ﾌｧｲﾙ(F)』→『**終了**(X)』と操作した場合、アプリケーションは終了する。

(5) 『ﾒﾆｭｰﾊﾞｰ』→『**ｵﾌﾞｼﾞｪｸﾄの挿入**』と操作した場合、「**ｵﾌﾞｼﾞｪｸﾄの挿入**」ダイアログボックスを表示して、任意のオブジェクトをフォームに表示する。

(6) 前述(2)～(5)の Menu コントロールの各メニュー項目は、次のとおりである。



(7) 「**ﾌｧｲﾙを開く**」「**ﾌｧｲﾙ名を付けて保存**」の各ダイアログボックスで キャンセル が Click された時、その処理を中止する。

(8) **ウィンドウ**をリサイズ操作した場合、**OLE コンテナコントロール**の大きさを調整する。

【実装】

(1) **例題 7-1** で作成した、フォーム、コントロール及びプロシージャを再利用する。

(2) **ツールボックス**上の CommonDialog を Click して、Form1 フォームに配置する。

(3) Menu コントロールを『ﾒﾆｭｰﾊﾞｰ』→『ﾌｧｲﾙ(F)』→『開く(O)』と操作して、コードウィンドウを開き、このメニュー項目に必要なイベントプロシージャを作成する。

① 「**ﾌｧｲﾙを開く**」ダイアログボックスで キャンセル の押下を検出するために、**エラートラッピング処理**を開始する。

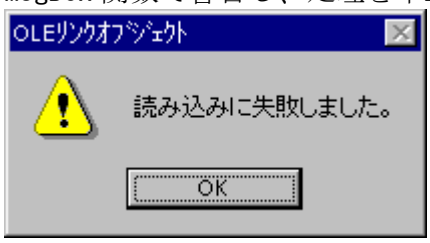
② **コモンダイアログボックス**の **CancelError プロパティ**に キャンセル の有効化を設定する。

③ 「**ﾌｧｲﾙを開く**」ダイアログボックスの[ﾌｧｲﾙ名]リストボックスに**OLE リンクオブジェクト**のデータファイル(拡張子は".OLE")だけを表示するために、**Filter プロパティ**に見出しとフィルタを設定する(テキスト･P568 と**応用 7** を参照)。

見出し	フィルタ	説　明
OLEﾌｧｲﾙ (*.OLE)	*.OLE	拡張子が"OLE"であるファイルだけを表示する。
すべて (*.*)	*.*	すべてのファイルを表示する。

【Memo】

④ 有効なデータファイルのパス及びファイル名を取得できなかった場合、次のとおり MsgBox 関数で警告し、処理を中止する。



〔④ パス及びファイル名が無効〕

〔⑩ キャンセル による読み込みの中止〕

⑤ 「ファイルを開く」ダイアログボックスの キャンセル の押下用のエラートラッピング処理を終了し、データファイルを読み込む時のエラートラッピング処理を開始する。

【Memo】

⑥ FreeFile 関数により、使用可能なファイル番号を取得する。詳細は、テキスト･P146 を参照。

⑦ **OPEN ステートメント**により、データファイルを**バイナリアクセス**形式で開く。詳細は、テキスト･P146 を参照。

⑧ **ReadFromFile メソッド**により、**OLE リンクオブジェクト**のデータファイルを読み込む。

【構文】

object.ReadFromFile *filenumber*

object　… 対象となるオブジェクトのオブジェクト式を指定。

filenumber　… オブジェクトを読み込む時に使うファイル番号を示す数式を指定。ファイル番号は、開いている**バイナリファイル**に対応していること。

解説：**SaveToFile メソッド**又は **SaveToOle1File メソッド**で保存した **OLE リンクオブジェクト**のデータファイルを読み込む。

【Memo】

⑨ 前述⑧で使用したファイルを閉じる。

⑩ 「ファイルを開く」ダイアログボックスで キャンセル が Click された時の**エラートラッピングルーチン**では、前述のとおり MsbBox 関数で警告し、処理を中止する。

(4) Menu コントロールを『メニューバー』→『ファイル(F)』→『**名前を付けて保存(S)**』と操作して、コードウィンドウを開き、このメニュー項目に必要なイベントプロシージャを作成する。

- 処理の流れは、前述の**実装**(3)とほぼ同様であるので参考にすること。
- 有効なパス及びファイル名が取得できない場合は、"**書き込みに失敗しました。**"と MsgBox 関数で表示する(前述の**実装**(3)④を参照)。
- キャンセル の Click により処理を中止する場合は、"**書き込みを中止しました。**"と MsgBox 関数で表示する(前述の**実装**(3)⑩を参照)。
- **OLE リンクオブジェクト**をデータファイルに書き込む場合は、**SaveToFile メソッド**を使う(詳細は、ヘルプを参照)。

【構文】

object.SaveToFile *filenumber*

object　　… 対象となるオブジェクトのオブジェクト式を指定。

filenumber　… オブジェクトを保存する時に使うファイル番号を示す数式を指定。
ファイル番号は、開いている**バイナリファイル**に対応していること。

解説：データファイルに**OLE リンクオブジェクト**を保存する。

【Memo】

(5) Menu コントロールを『ﾒﾆｭｰﾊﾞｰ』→『ﾌｧｲﾙ(F)』→『終了(X)』と操作して、コードウィンドウを開き、このメニュー項目に必要なイベントプロシージャを作成する。

(6) Menu コントロールを『ﾒﾆｭｰﾊﾞｰ』→『ｵﾌﾞｼﾞｪｸﾄの挿入』と操作して、コードウィンドウを開き、このメニュー項目に必要なイベントプロシージャを作成する。

【課題】

(1) "D:¥Home¥User1XX¥VB¥Reidai7"配下に、OLE **リンクオブジェクト**のデータファイルを作成しなさい。ただし、保存するデータファイル名は"OLE_Rei1.OLE"とし、**挿入オブジェクト**は"¥¥XXXXXXXX¥Public¥Microsoft Visual Basic¥Example¥Seiseki1.XLS"とする。

【Memo】

(2) CommandPrompt を起動して、TYPE コマンドと VIEW.EXE により上記(1)で作成したデータファイルの内容を確認しなさい。

【Memo】

(3) 次の操作を行い、その動作・結果を確認しなさい。

【手順】

① 『ﾒﾆｭｰﾊﾞｰ』→『ｵﾌﾞｼﾞｪｸﾄの挿入』→『ﾌｧｲﾙから(F)』と操作し、[参照(B)...] を Click する。

②「参照」ダイアログボックスで、次のファイルを選択し、[開く(O)] を Click する。
"¥¥OkaGp200¥Public¥Microsoft Visual Basic¥Example¥Seiseki1.XLS"

③「ﾌｧｲﾙ名(E):」のファイル名と、「ﾘﾝｸ(L):」を確認して、[OK] を Click する。

④ 『ﾒﾆｭｰﾊﾞｰ』→『名前を付けて保存(S)』と操作し、次のファイル名で保存する。
"D:¥Home¥User1XX¥VB¥Reidai7¥OLE_Rei2.OLE"

⑤ 『ﾒﾆｭｰﾊﾞｰ』→『ﾌｧｲﾙを開く(O)』と操作し、前述④のファイルを開く。

⑥ **サーバアプリケーション**側で、データを操作してみる。

【Memo】

(4) CommandPrompt を起動して、TYPE コマンドと VIEW.EXE により上記(3)で作成したデータファイルの内容を確認しなさい。また、上記(1)のデータファイルと比較しなさい。

【Memo】

【考察】