Ｖｉｓｕａｌ　Ｂａｓｉｃ

（№２）

科　　名		氏　　名	
Tutorial group			

2004.6

１．Sub プロシージャの宣言

【*すぐ解決(3)*】

［ファイル名］

D:¥Home¥User1XX¥VB¥BB¥P110(P110.VBP, P110.FRM)

［プログラムの説明］

テキスト･P110 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］

［プログラム］

```
Option Explicit
Private Sub Command1_Click()
    Dim intMax As Integer       ' ローカル変数の宣言
    intMax = 123
① ──▶ subProc (intMax)          ' 値渡しでプロシージャを呼び出す
    MsgBox Str(intMax), vbInformation, "引数の値渡し"
End Sub
Private Sub Command2_Click()
    Dim intMax As Integer       ' ローカル変数の宣言
    intMax = 123
    CountFiles                  ' 引数無しでプロシージャを呼び出す
    CountFiles (intMax)         ' 引数有りでプロシージャを呼び出す
End Sub
② ──▶Private Sub subProc(ByVal intMax As Integer)
    intMax = 987                ' ローカル変数に整数定数を代入
    MsgBox Str(intMax), vbInformation, "sub プロシージャ"
End Sub
Private Sub CountFiles(Optional intMaxFile As Variant)
    If IsMissing(intMaxFile) Then   ' IsMissing 関数により実引数の有無を確認
        ' intMaxFile が渡されなかった
        MsgBox ("何か忘れていませんか")
    Else
        MsgBox Str(intMaxFile), vbInformation, "引数がオプション"
    End If
End Sub
```

【考察】

矢線②の ByVal キーワードを取り除いて、**「参照渡し」**になることを確認しなさい。

２．Function プロシージャの宣言

【*すぐ解決(4)*】

［ファイル名］

D:¥Home¥User1XX¥VB¥BB¥P112(P112.VBP, P112.FRM)

［プログラムの説明］

テキスト･P112 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］

"Print メソッド"を使用して、フォームの**クライアント領域**に直接出力する。

［プログラム］

```
Option Explicit
Private Sub Command1_Click()
    Dim intResult As Integer
    Cls                             ' フォームのクライアント領域をクリア
    intResult = Add1(5)             ' Function プロシージャの呼び出し
    Print "結果は " & Format(intResult, " ##0")    ' フォームに出力
    intResult = Add1(15)            ' Function プロシージャの呼び出し
    Print "結果は " & Format(intResult, " ##0")    ' フォームに出力
End Sub
Function Add1(intAdd1ToMe As Integer) As Integer
    Add1 = intAdd1ToMe + 1          ' Function 名自身に値を代入
End Function
```

【考察】

［ファイル名］

D:¥Home¥User1XX¥VB¥BB¥P116(P116.VBP, P116.FRM)

［プログラムの説明］

テキスト･P116 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］

［プログラム］

```
Option Explicit
Private Sub Command1_Click()
    Dim intResult As Integer
    intResult = Counter1()
```

```
    MsgBox "結果は " & Str(intResult), , "Dim 宣言"
End Sub
Function Counter1() As Integer
    Dim intCountValue As Integer' ローカル変数の宣言
    intCountValue = intCountValue + 1
    Counter1 = intCountValue    ' Function 名自身に値を代入
End Function
Private Sub Command2_Click()
    Dim intResult As Integer
    intResult = Counter2()
    MsgBox "結果は " & Str(intResult), , "Static 宣言"
End Sub
Function Counter2() As Integer
    Static intCountValue As Integer ' 静的変数の宣言
    intCountValue = intCountValue + 1
    Counter2 = intCountValue    ' Function 名自身に値を代入
End Function
```

【考察】

3．Visual Basic のコントロール

【*すぐ解決(5)*】

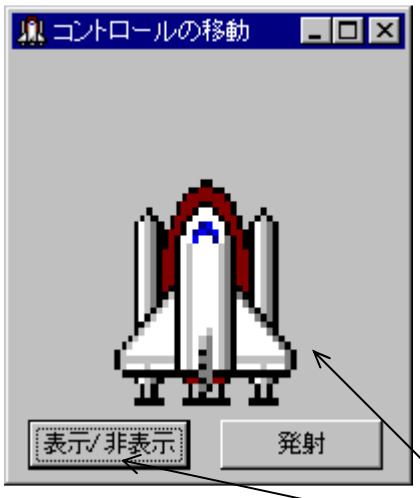
［ファイル名］

D:¥Home¥User1XX¥VB¥BB¥P136(P136.VBP, P136.FRM, P136.FRX)

［プログラムの説明］

Visual Basic のコントロールの操作に関する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］



プロパティ名	設定値
Form1.Height	3600
Form1.Width	3060
Image1.Height	1695
Image1.Left	720
Image1.Picture	¥¥OkaGp200¥Public¥Microsoft Visual Basic¥Graphics¥Icons¥Intustry¥ROCKET.ICO
Image1.Top	960
Image1.Width	1455

［プログラム］

```
Option Explicit
Private Sub Command1_Click()
```

```
    Static flag As Boolean      ' 論理型静的変数の宣言
    If flag Then
        Image1.Visible = True   ' イメージコントロールを表示
    Else
        Image1.Visible = False  ' イメージコントロールを非表示
    End If
    flag = Not flag             ' 現在の flag の否定を代入
    Image1.Top = 960            ' イメージコントロールを初期状態にする
End Sub
Private Sub Command2_Click()
    Dim intCNT As Integer
    For intCNT = 960 To 0 Step -1
        Image1.Move 720, intCNT ' Move メソッドによるコントロールの移動
    Next intCNT
End Sub
```

【課題 1】

次の機能を追加実装しなさい。

(1) コマンドボタンの表示(Caption)が、ロケットのイメージが表示されている時は"**非表示**"に、イメージが表示されていない時は"**表示**"に切り替わる。

(2) ロケットのイメージが下方(初期状態)にある時にのみ、"**発射**"ボタンが有効になる。

【考察】

.....

.....

.....

.....

【課題 2】

前述・**【課題 1】**の機能に、さらに次の機能を追加実装しなさい。

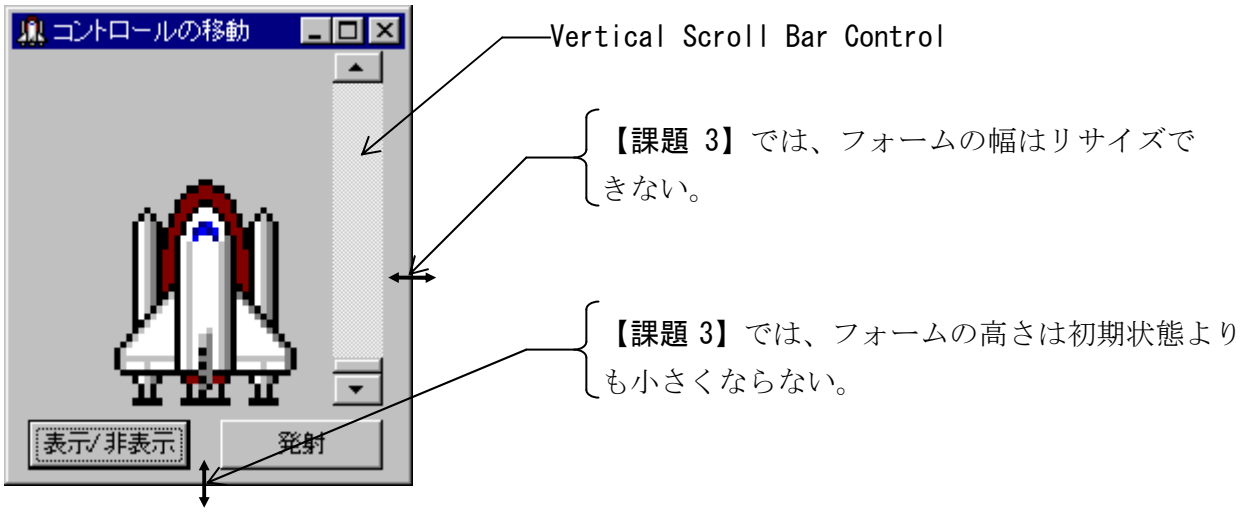
(1) 次の[フォーム]のように、VScrollBar を追加しなさい。

(2) VScrollBar を操作すると、ロケットのイメージが上下する。

[ファイル名]

D:¥Home¥User1XX¥VB¥BB¥P136A(P136A.VBP, P136A.FRM, P136A.FRX)

[フォーム]



【考察】

【課題 3】

前述・**【課題 2】**の機能に、フォームをリサイズした場合の次の機能を追加実装しなさい。

(1) フォームの幅は、直ちに初期状態に戻る。
⇒ イベントプロシージャ内で、常に Me.Width プロパティに初期値を直接代入する。

(2) フォームの高さは、初期状態よりも小さくはならない。
⇒ イベントプロシージャ内で、Me.Height プロパティの値が初期値よりも小さくなった場合は、このプロパティに初期値を直接代入する。

(3) フォームの高さを大きくした場合、コマンドボタンコントロールとイメージコントロールはフォームの下端からの位置関係は常に初期状態と同じである。
⇒ イベントプロシージャ内で、各コントロールの Top プロパティに Me.Height プロパティからの差を代入する。

(4) スクロールバーコントロールの高さは、フォームの高さに合わせる。
⇒ イベントプロシージャ内で、VScroll1.Top プロパティに Me.Height プロパティからの差を代入する。

(5) スクロールバーコントロールの高さに合わせて、Max プロパティ値を調整する。
⇒ イベントプロシージャ内で、VScroll1.Max プロパティに Me.Height プロパティからの差を代入する。

(6) スクロールバーコントロールの高さに合わせて、Value プロパティ値を調整する。
⇒ イベントプロシージャ内で、VScroll1.Value プロパティに VScroll1.Max プロパティ値を代入する。

【考察】

4．[開く]ダイアログボックスと[名前を付けて保存]ダイアログボックス

[開く]ダイアログボックスと**[名前を付けて保存]ダイアログボックス**を使用する場合は、次の手順により**ツールボックス**に**コントロール**を追加しておく。

【手順】

(1) [**ツールボックス**]を右クリックする。
(2) [ｺﾝﾎﾟｰﾈﾝﾄ(O)...]をクリックする。
(3) [ｺﾝﾎﾟｰﾈﾝﾄ]ダイアログボックスの、[ｺﾝﾄﾛｰﾙ]タブを選択する。
(4) [☐Microsoft Common Dialog Control 5.0]をチェックする。
(5) [OK]をクリックする。
(6) [**ツールボックス**]に  が追加されたのを確認する。

【*すぐ解決(6)*】

[ファイル名]

D:¥Home¥User1XX¥VB¥BB¥P145(P145.VBP, P145.FRM, P145.FRX)

[プログラムの説明]

テキスト･P145 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、事前に Microsoft Common Dialog Control 5.0 を追加しておくこと。

[フォーム]

[プログラム]

```
Option Explicit
Private Sub Command1_Click()
    ' Microsoft Common Dialog Control 5.0(SP2)を追加
① ──→ On Error GoTo Cancel                  ' エラーハンドラ(ｴﾗｰﾄﾗｯﾋﾟﾝｸﾞﾙｰﾁﾝ)の設定
    CommonDialog1.ShowOpen  ←──── ②         ' Common Dialog を呼び出す
    MsgBox "次のファイルを開く: " & CommonDialog1.filename
③ →Cancel:                                  ' エラーハンドラ(ｴﾗｰﾄﾗｯﾋﾟﾝｸﾞﾙｰﾁﾝ)の入口
End Sub
```

【解説】

Visual Basic では、プログラムの実行中にエラーが発生した場合、プログラムの実行を停止することなく､このエラーを捕捉して**割り込み処理(interrupt/trap)**によって任意の処理を実行させることができる。

この任意の処理を、**エラーハンドラ(error handler:エラートラッピングルーチン(error trapping routine))**と言う。

なお、**エラーハンドラ**の設定(①)と**エラーハンドラ**の入口(entry point)(③)は、そのプロシージャ内にのみ記述できる。

エラーが発生した時のエラー情報の取得､エラーが発生した時に分岐する**トラッピングルーチン**(trapping routine)を設定する *statement*、関数及びオブジェクトを、次に示す。

Erl()	エラーが発生した行番号の取得
Err()	エラーコードの取得
Err	Err()の戻り値を設定
Error	エラーを擬似的に発生
Error()	指定したエラー番号に対応するエラーメッセージの取得
On Error	エラーハンドラ(ﾄﾗｯﾋﾟﾝｸﾞﾙｰﾁﾝ)の設定
Resume	エラーハンドラ(ﾄﾗｯﾋﾟﾝｸﾞﾙｰﾁﾝ)からの復帰
Err	エラー情報を持つオブジェクト

詳細は後述する。

【考察】

【課題1】

[開く]**ダイアログボックス**から、パス名を除いたファイル名だけを MsgBox 関数を使用して表示しなさい。

【考察】

filename プロパティ

FileTitle プロパティ

ShowOpen メソッド

ShowSave メソッド

［ファイル名］

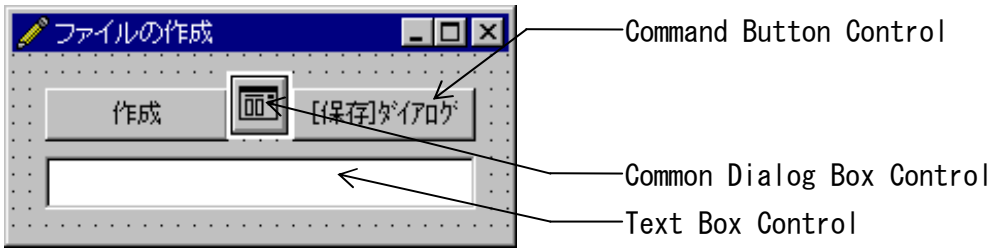
D:¥Home¥User1XX¥VB¥BB¥P146(P146.VBP, P146.FRM, P146.FRX)

［プログラムの説明］

テキスト･P146 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、事前に"D:¥Home¥User1XX¥VB¥TMP"フォルダを作成しておくこと。また、ディスクアクセスを伴うので、プログラミングは慎重に行うこと。

［フォーム］



［プログラム］

```
Option Explicit
Private Sub Command1_Click()
① ⟶ On Error GoTo FileError          ' エラーハンドラの設定
     Open "D:¥Home¥User1XX¥VB¥TMP¥file.txt" For Output As #1 ' ファイルを開く
     Print #1, Text1.Text             ' ファイルへ書き込む
     Close #1                         ' ファイルを閉じる
② ⟶ Exit Sub                         ' Sub プロシージャを中止する
     ' 正常に処理される場合、以降のエラーハンドラを実行する必要が無い。
③ ⟶FileError:                        ' エラーハンドラの入口
     MsgBox "ファイルエラー", vbCritical, "ファイルの作成"
End Sub
Private Sub Command2_Click()
     ' Microsoft Common Dialog Control 5.0(SP2)を追加
     On Error GoTo Cancel
     CommonDialog1.ShowSave
     MsgBox "次のファイルに保存:" & CommonDialog1.filename, vbInformation, "[名前を付けて保存]ﾀﾞｲｱﾛｸﾞ"
Cancel:
End Sub
```

【考察】

出力結果を、「メモ帳」で確認すること。

Print#ステートメントで書き込んだ内容は、**Line Input#ステートメント**で読み出すことができる。

［ファイル名］

D:¥Home¥User1XX¥VB¥BB¥P148(P148.VBP, P148.FRM, P148.FRX)

［プログラムの説明］

テキスト･P148 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、事前に"D:¥Home¥User1XX¥VB¥TMP"フォルダを作成しておくこと。また、ディスクアクセスを伴うので、プログラミングは慎重に行うこと。

［フォーム］

［プログラム］

```
Option Explicit
Private Sub Command1_Click()
    On Error GoTo FileError          ' エラーハンドラの設定
    Open "D:¥Home¥User1XX¥VB¥TMP¥data.dat" For Output As #1     ' ファイルを開く
①→ Write #1, Val(Text1.Text), Val(Text2.Text), Val(Text3.Text) ' ファイルへ書き込む
    Close #1                         ' ファイルを閉じる
    Exit Sub                         ' Sub プロシージャを中止する
    ' 正常に処理される場合、以降のエラーハンドラを実行する必要が無い。
FileError:                           ' エラーハンドラの入口
    MsgBox "ファイルエラー : " & Err & " " & Error(Err)
End Sub
```

【考察】

出力結果を、「メモ帳」で確認すること。

Write#ステートメントで書き込んだ内容は、**Input#ステートメント**で読み出すことができる。

Err()関数

Error()関数

【課題 2】

矢線①の **Write#ステートメント**を、**Print#ステートメント**に書き換えて実行し、その出力結果を「メモ帳」で確認しなさい。

また、**Write#ステートメント**と **Print#ステートメント**の相違を具体的に考察しなさい。

[ファイル名]

D:¥Home¥User1XX¥VB¥BB¥P149(P149.VBP, P149.FRM, P149.FRX)

[プログラムの説明]

テキスト･P149 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、事前に"D:¥Home¥User1XX¥VB¥TMP"フォルダを作成しておくこと。また、ディスクアクセスを伴うので、プログラミングは慎重に行うこと。

[フォーム]

[プログラム]

```
Option Explicit
Dim FileNumber As Integer           ' モジュール変数の宣言
Private Sub Command1_Click()
    On Error GoTo OpenError         ' エラーハンドラの設定
    FileNumber = FreeFile           ' 使用可能なファイル番号を取得
    Open "D:¥Home¥User1XX¥VB¥TMP¥data.dat" For Output As #FileNumbe ' ファイルを開く
    On Error GoTo WriteError        ' エラーハンドラの設定
    Write #FileNumber, Val(Text1.Text), Val(Text2.Text), Val(Text3.Text)
    Close #FileNumber               ' ファイルを閉じる
    Exit Sub                        ' Sub プロシージャを中止する
    ' 正常に処理される場合、以降のエラーハンドラを実行する必要が無い。
OpenError:                          ' エラーハンドラの入口
    MsgBox "ファイルオープンエラー", vbCritical, "例題 P149"
    Close #FileNumber               ' ファイルを閉じる
    Exit Sub                        ' Sub プロシージャを中止する
WriteError:                         ' エラーハンドラの入口
    MsgBox "ファイル書き込みエラー", vbExclamation, "例題 P149"
    Close #FileNumber               ' ファイルを閉じる
End Sub
```

```
Private Sub Command2_Click()
    Dim int1, int2, int3 As Integer ' ローカル変数の宣言

    On Error GoTo FileError         ' エラーハンドラの設定
    Err = 0                         ' Err()関数の戻り値の初期化
    FileNumber = FreeFile           ' 使用可能なファイル番号を取得
    Open "D:¥Home¥User1XX¥VB¥TMP¥data.dat" For Input As #FileNumber ' ファイルを開く
    Input #FileNumber, int1, int2, int3
    Text4.Text = Str(int1)
    Text5.Text = Str(int2)
    Text6.Text = Str(int3)
    Close #FileNumber               ' ファイルを閉じる
    Exit Sub                        ' Sub プロシージャを中止する
    ' 正常に処理される場合、以降のエラーハンドラを実行する必要が無い。
FileError:                          ' エラーハンドラの入口
    Select Case Err
        Case 52
            MsgBox "ファイル名または番号が不正", vbCritical, "例題 P149"
        Case 53
            MsgBox "ファイルオープンエラー", vbCritical, "例題 P149"
        Case 54, 62
            MsgBox "ファイル読み込みエラー", vbExclamation, "例題 P149"
    End Select
    Close #FileNumber
End Sub
```

【考察】

FreeFile()関数

[ファイル名]

D:¥Home¥User1XX¥VB¥BB¥P151(P151.VBP, P151.FRM, P151.FRX)

[プログラムの説明]

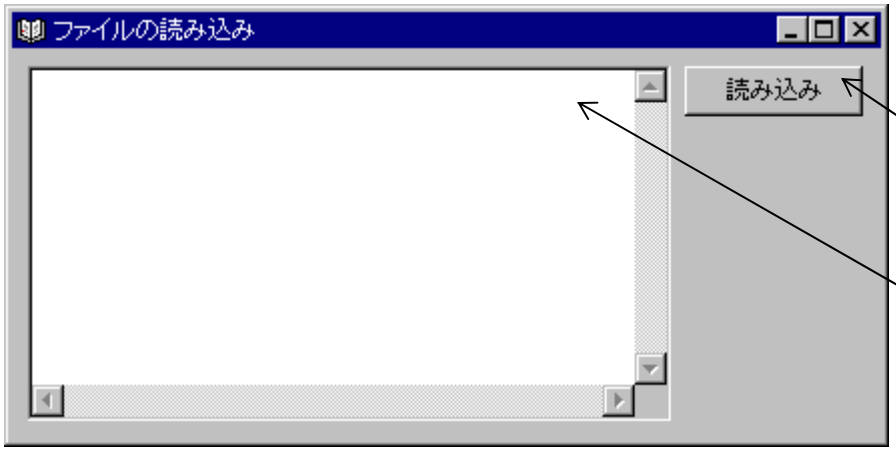
テキスト･P151 のプログラムを次のように作成し、その動作を確認しなさい。

なお、事前に"D:¥Home¥User1XX¥VB¥TMP"フォルダを作成しておくこと。また、ディスクアクセスを伴うので、プログラミングは慎重に行うこと。

次のファイルを、各自のフォルダにコピーして置くこと。

¥¥XXXXXXXX¥Public¥Microsoft Visual Basic¥file.txt

[フォーム]



[プログラム]

```
Option Explicit
Private Sub Command1_Click()
    Dim NewLine As String
    Dim FileNumber As Integer

    On Error GoTo FileError          ' エラーハンドラの設定
    FileNumber = FreeFile            ' 使用可能なファイル番号を取得
    Open "D:¥Home¥User1XX¥VB¥TMP¥file.txt" For Input As #FileNumber
    Do Until EOF(FileNumber)         ' ファイルの終端になるまで処理を繰り返す
①       Line Input #FileNumber, NewLine              ' ファイルから１行を読み込む
        Text1.Text = Text1.Text & NewLine & vbCrLf   ' TextBox に１行を追加出力する
    Loop
    Close #FileNumber                ' ファイルを閉じる
    Exit Sub                         ' Sub プロシージャを中止する
    ' 正常に処理される場合、以降のエラーハンドラを実行する必要が無い。
FileError:                           ' エラーハンドラの入口
    MsgBox "ファイルエラー", vbExclamation, "例題 P151"
    Close #FileNumber
End Sub
```

【考察】

矢線①の **Line Input#ステートメント**を、**Input#ステートメント**に書き換えて実行し、その出力結果を「**ﾒﾓ帳**」で確認しなさい。

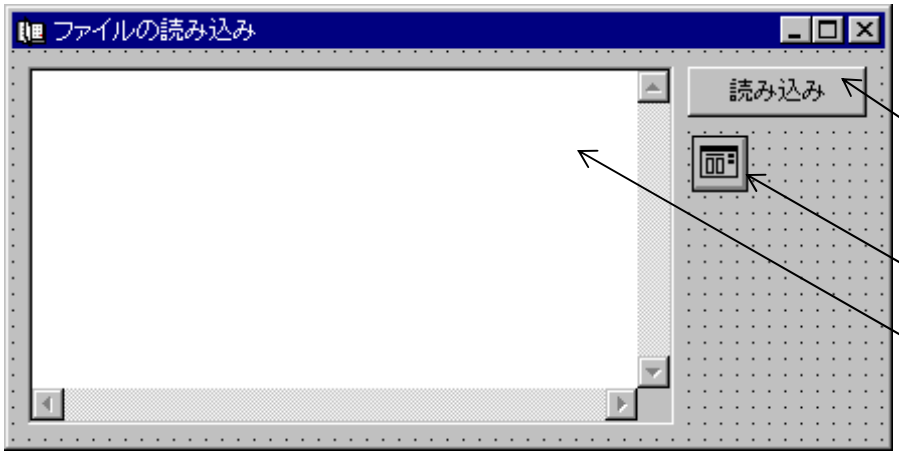
EOF()関数

vbCrLf

【課題 3】

テキスト･P151 のプログラムを、任意のファイルを読み込むプログラムに変更しなさい。

[フォーム]



【考察】

[ファイル名]

D:¥Home¥User1XX¥VB¥BB¥P152(P152.VBP, P152.FRM, P152.FRX)

[プログラムの説明]

テキスト･P152 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、事前に"D:¥Home¥User1XX¥VB¥TMP"フォルダを作成しておくこと。また、ディスクアクセスを伴うので、プログラミングは慎重に行うこと。

[フォーム]

[プログラム]

```
Option Explicit
Private Sub Command1_Click()
    Dim FileNumber As Integer       ' ローカル変数の宣言

    On Error GoTo FileError         ' エラーハンドラの設定
```

```
    FileNumber = FreeFile           ' 使用可能なファイル番号を取得
    Open "D:¥Home¥User1XX¥VB¥TMP¥file.txt" For Input As #FileNumber ' ファイルを開く
    Text1 = Input$(LOF(FileNumber), #FileNumber)  ' ファイル全体を読み込む
    Close #FileNumber               ' ファイルを閉じる
    Exit Sub                        ' Sub プロシージャを中止する
    ' 正常に処理される場合、以降のエラーハンドラを実行する必要が無い。
FileError:                          ' エラーハンドラの入口
    MsgBox "エラーコード：" & Err & vbCr & Error(Err), vbCritical, "例題 P152"
    Close #FileNumber               ' ファイルを閉じる
End Sub
```

【考察】

LOF()関数

FileLen()関数

◆ **シーケンシャルアクセス形式**のまとめ

(1) オープンモード

Input	ファイルがシーケンシャルアクセスの入力モードでオープンされる。
Output	ファイルがシーケンシャルファイルへの出力モードでオープンされる。
Append	ファイルがシーケンシャルファイルへの追加モードでオープンされる。

(2) アクセスステートメント

アクセス単位	読み込み用	書き込み用
1行単位	Line Input#	Print#
1項目単位	Input#	Write#
その他	Input()	

(3) ファイルアクセスに関する関数

EOF()	ファイルのアクセス位置が、ファイルの終端に達したか否かを論理型で返す。
LOF()	オープンしたファイルのサイズをバイト単位で返す。
FileLen()	指定したファイルのサイズをバイト単位で返す。

例題 5-1 次の機能の説明を読んで、順編成ファイルを読み込み、画面に表示するアプリケーションを作成しなさい。


D:¥Home¥User1XX¥VB¥Reidai5


Project1(ReadIn.vbp)


Form1(ReadIn.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 5-1 行単位の読み込み
アイコン	FILES07.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(読み込み)を Click すると、次の順編成ファイルを 1 行読み込んで、TextBox コントロールに表示する。
順編成ファイルの仕様は、次のとおりとする。

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¥Reidai5¥Rei-5-1.TXT

(2) ファイルが終了した場合、CommandButton コントロール(読み込み)を無効にする。

(3) 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。

【解説】

(1) Visual Basic が取り扱うファイルのアクセス方式
① **シーケンシャルアクセス**
② **ランダムアクセス**
} データフォーマットに依存
③ **バイナリアクセス** … バイト単位でアクセス

(2) ファイルを取り扱う手順
① ファイルをオープンする。

【構文】

Open *pathname* **[For** *mode***] [Access** *access***] [***lock***] As [#]** *filenumber* **[Len=***reclength***]**

pathname　… ファイル名を指定。絶対パスでも指定できる。

mode　… ファイルモードを指定。

access　… ファイルの処理方法を指定。

lock　… 他のプロセスからのアクセスの制御を指定。

filenumber　… ファイル番号を指定。

reclength　… レコード長を指定。

・ Open *pathname* For Input As #1
・ Open *pathname* For Random As #1 Len = *length*
・ Open *pathname* For Binary Access Read As #1

② 変数を介して、ファイルへアクセスする。

・ Line Input #1, *TextLine*
・ Input #1, *MyRec1*, *MyRec2*
・ *MyChar* = InputB(1, #1)

③ ファイルをクローズする。

【構文】

Close [*filenumberlist***]**

filenumberlist　… ファイル番号を指定。

(3) ファイル操作に関する Visual Basic の主なステートメントと関数

アクセス方式	ステートメント		関数	
ｼｰｹﾝｼｬﾙｱｸｾｽ (Sequential Access)	Open	Close	Input	InputB
	Input#	Line Input#	Dir	EOF
	Print#	Write#	Loc	LOF
			FileCopy	FileDataTime
			FileLen	FreeFile
			GetAttr	SetAttr
			Seek	
ﾗﾝﾀﾞﾑｱｸｾｽ (Random Access)	Open	Close	同上	
	Get	Put		
ﾊﾞｲﾅﾘｱｸｾｽ (Binary Access)	Open	Close	同上	
	Get	Put		
	Input	InputB		

(4) MS-DOS 系のテキストファイルでは、行の区切りに **Chr(&HD)+Chr(&HA)** が使用されている。また、ファイルの終了は、Visual Basic では EOF()関数で調べる。

【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　: 3 - 両方

(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) 読み込むファイル名は、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。

(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(読み込み)の処理は、次のとおりである。

(a) ファイルの読み込み位置が、末尾に到達していない場合

- Click する毎にファイルから 1 行読み込んで、TextBox コントロールに表示する。
- 1 行読み込みには、Line Input#ステートメントを使用する。

(b) ファイルが終了した場合

- CommandButton コントロール(読み込み)を、無効にする。
- コントロールを無効にするには、Enabled プロパティを操作する。

(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 関数で次のとおりに表示する。

prompt: "終了します。"

buttons: 

title: "例題５－１入力用"

戻り値: なし

応用 5-1 例題 5-1 のアプリケーションを、次のとおりに変更しなさい。

【機能】

(1) CommandButton コントロール(読み込み)を Click すると、例題の順編成ファイルをすべて読み込んで、TextBox コントロールに表示する。

(2) ファイルが終了した場合、ファイルをクローズして、CommandButton コントロール(読み込み)を無効にする。

【実装】

(1) CommandButton コントロール(読み込み)の処理は、次のとおりに変更する。

(a) ファイルの読み込み位置が、末尾に到達していない場合

- ファイルが終了するまで、1 行ずつ読み込んで TextBox コントロールに表示する処理を繰り返す。

(b) ファイルが終了した場合

- ファイルをクローズする。
- CommandButton コントロール(読み込み)を、無効にする。

(2) CommandButton コントロール(終了)でアプリケーションを終了する時、前述･実装(7)のとおりに MsgBox 関数で表示する。

【考察】

例題 5-2 次の各コントロールを確認するアプリケーションを作成しなさい。

D:¥Home¥User1XX¥VB¥Reidai5

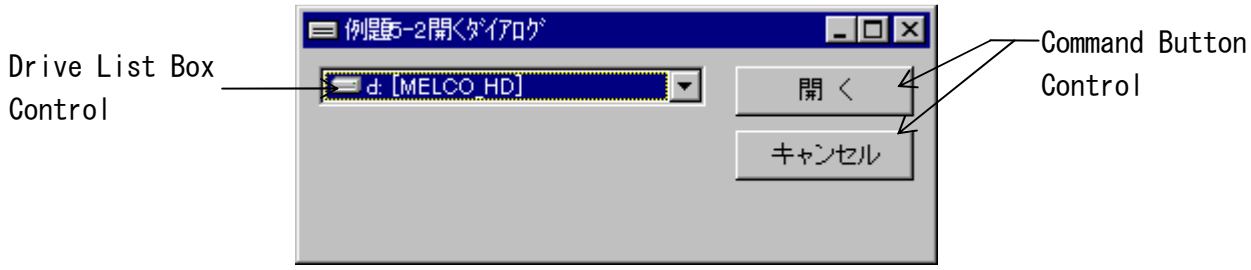
Project1(OpenDLG.vbp)

Form1(OpenDLG.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 5-2 開くﾀﾞｲｱﾛｸﾞ
アイコン	DRIVE01.ICO
Picture のセットアップされている path	
¥¥OkaGp200¥Public¥Microsoft Visual Basic¥GRAPHICS¥ICONS¥COMPUTER	

〔アプリケーションの外観〕



【機能】

(1) DriveListBox コントロールを操作すると、接続されているドライブを選択する。

(2) CommandButton コントロール(開　く)を Click すると、選択されたドライブ名を MsgBox 関数で表示する。

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

【実装】

(1) DriveListBox コントロールを、Form1 フォームに配置する。

(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) 前述･機能(2)の MsgBox 関数の表示は、次のとおりである。

prompt:　選択されたドライブ名

buttons:

title:　"例題５－２開く"

戻り値:　なし

【考察】

DriveListBox コントロールの主なイベントプロシージャとプロパティを調査しなさい。

Change

Drive

【追加機能①】

(1) DirListBox コントロールを操作すると、カレントドライブ内の任意のサブディレクトリを選択する。⇒  Directory List Box Control

(2) CommandButton コントロール(開 く)を Click すると、選択されたサブディレクトリの絶対パスを MsgBox 関数で表示する。

【実装】

(1) DirListBox コントロールを、Form1 フォームに追加配置する。

(2) DriveListBox コントロールを操作した場合、選択されたドライブ名を DirListBox コントロールに設定する。

【構文】

Left(*string, length*)

string … 取り出す元の文字列式。

length … 取り出す文字数を示す数式を指定。

戻り値 … 取り出された文字列。

【構文】

Right(*string, length*)

string … 取り出す元の文字列式。

length … 取り出す文字数を示す数式を指定。

戻り値 … 取り出された文字列。

【構文】

Mid(*string, start[, length]*)

string … 取り出す元の文字列式。

start … *string*の先頭の位置を 1 として、取り出す文字列の先頭位置を指定。

length … 取り出す文字数を示す数式を指定。

戻り値 … 取り出された文字列。

【考察】

DirListBox コントロールの主なイベントプロシージャとプロパティを調査しなさい。

Change

Path

【追加機能②】

(1) FileListBox コントロールを操作すると、カレントディレクトリ内の任意のファイルを選択する。⇒  File List Box Control

(2) CommandButton コントロール(開 く)を Click すると、選択されたファイルの**絶対パス**とファイル名を MsgBox 関数で表示する。

【実装】

(1) FileListBox コントロールを、Form1 フォームに追加配置する。

(2) DirListBox コントロールを操作した場合、選択されたサブディレクトリの path を FileListBox コントロールに設定する。

【考察】

FileListBox コントロールの主なイベントプロシージャとプロパティを調査しなさい。

Click

DblClick

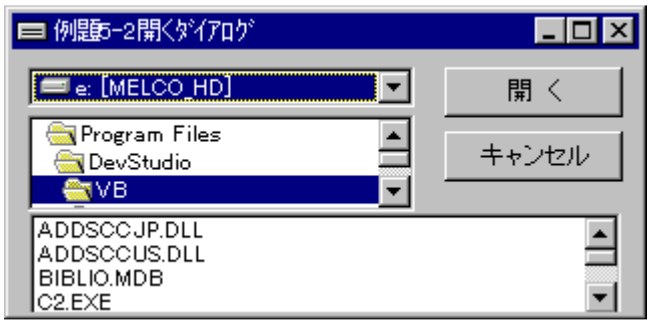
filename

Path

【追加機能③】

(1) CommandButton コントロール(開 く)を Click すると、選択されたファイルの**絶対パス**とファイル名及びファイルの属性を MsgBox 関数で表示する。

〔アプリケーションの外観〕



【実装】

(1) ファイルの属性は、GetAttr()関数で取得する。

【構文】

GetAttr(*pathname*)

pathname　　… ファイル名又はフォルダ名を指定。**絶対パス**でも指定できる。

戻り値　　… ファイルまたはフォルダの属性を表す整数値の合計値。

定数	値	内容	定数	値	内容
vbNormal	0	通常ファイル	vbSystem	4	システムファイル
vbReadOnly	1	読取り専用ファイル	vbDirectory	16	フォルダ
vbHidden	2	隠しファイル	vbArchive	32	アーカイブ

(2) CommandButton コントロール(開 く)を Click した場合の MsgBox 関数の表示例は、次のとお

りである。

〔MsgBox 関数の表示例〕

【考察】

【追加機能④】

(1) CommandButton コントロール(開 く)を Click した場合、エラーの発生が予測されるので、その**エラー処理(エラートラッピング)**を行う。

(2) 前述(1)のエラーの一つに、**絶対パス**のエラーが予測されるが、それをあらかじめ回避する。

【解説】

(1) **エラー処理(エラートラッピング)**の作成手順は、次のとおりである。

① **エラーハンドラ**の設定

エラーの発生が予測されるステートメントの直前に、次のステートメントを記述する。

【構文】

On Error GoTo *line* **| Resume Next | GoTo 0**

GoTo *line* … エラーが発生した場合の、分岐先エラー処理ルーチンを有効にする。この分岐先エラー処理ルーチンは、*line* で指定する。

Resume Next … エラーが発生してもプログラムを中断せず、エラーが発生したステートメントの次のステートメントから実行を継続する。

GoTo 0 … エラー処理(エラートラッピング)を無効にする。

② **エラートラップ**

エラーが発生すると、**On Error ステートメント**で設定された方法によりエラー処理する。**GoTo ステートメント**によりエラー処理ルーチンが指定されている場合、*line* で指定された**エラー処理ルーチン(エラートラッピングルーチン)**に制御を渡す。

③ エラー処理

予測されるエラーに対する処理の記述と、エラー処理終了後のプログラムの実行の再開ステートメントを指定する。

【構文】

line:

line… プログラムコードの各行を識別するために記述する。
同一モジュール内では、**行ラベル**の重複は不可。

【構文】

Resume [0] | Next | *line*

Resume [0] … エラーの原因となったステートメントからプログラムの実行を再開。

Resume Next … エラーの原因となったステートメントの次のステートメントからプログラムの実行を再開。

Resume *line* … *line*に指定した行からプログラムの実行を再開。

(2) DirListBox コントロールの Path プロパティの値は、次のように設定される。

(a) ルートディレクトリ … **ドライブレター：￥**

(b) サブディレクトリ … **ドライブレター：￥サブディレクトリ名**

※ **絶対パス**を表す文字列の最後に"¥"が付加されたり、付加されなかったりする。

【実装】

(1) CommandButton コントロール(開 く)のイベントプロシージャ内に、エラーハンドラの設定とエラー処理ルーチンを追加記述する。

(2) エラー処理ルーチンの仕様は、次のとおりである。

ErrHandler: ⇦ **行ラベル**。エラー処理ルーチンの entry point

エラー番号とエラーメッセージを MsgBox 関数で表示する。

Resume Next

(3) エラー番号とエラーメッセージは、**Err オブジェクト**から取得する。

【構文】

Err[.*method*] | [.*property*]

解説：Err オブジェクトは、実行時エラーに関する情報を保有している。

Number プロパティ … 有効なエラー番号が格納される。

Descriptionプロパティ… エラー番号に対応したエラーメッセージが格納される。

Clear メソッド … **Err オブジェクト**のすべてのプロパティの設定値をクリア。

Raise メソッド … 実行時エラーを生成する。

(4) **絶対パス**を表す文字列の最後に、"¥"が付加されているかを確認する関数プロシージャを記述する。仕様は次のとおりである。

CheckPath(Byval sPath As String) As String

sPath: 任意のファイルを識別するための**絶対パス**

戻り値: **絶対パス**を表す文字列の最後に、必ず"¥"が付加された文字列

※ Hint:**絶対パス**を表す文字列の最後の 1 文字を取り出し、その文字が"¥"でない場合のみ文字列の最後に"¥"を付加する。

【考察】

応用 5-2 例題 5-2 のアプリケーションを、次のとおりに変更しなさい。


D:¥Home¥User1XX¥VB¥Reidai5

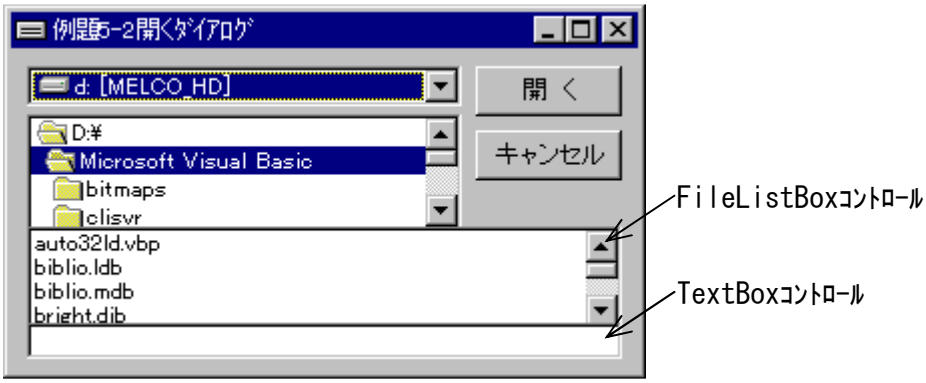

Project1(OpenDLG.vbp)


Form1(OpenDLG.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	応用 5-2 開くﾀﾞｲｱﾛｸﾞ
アイコン	DRIVE01.ICO
Picture のセットアップされている path	
¥¥OkaGp200¥Public¥Microsoft Visual Basic¥GRAPHICS¥ICONS¥COMPUTER	

〔アプリケーションの外観〕



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¥OpenDLG.EXE

【機能】

(1) DriveListBox コントロールを操作すると、接続されているドライブを選択できる。

(2) DirListBox コントロールを操作すると、カレントドライブ内の任意のサブディレクトリを選択する。
また、DriveListBox コントロールが操作された場合、DirListBox コントロールも自動的に更新される。

(3) FileListBox コントロールを操作すると、カレントディレクトリ内の任意のファイルを選択する。なお、FileListBox コントロールの操作は、次のとおりである。
① DirListBox コントロールが操作された場合、FileListBox コントロールも自動的に更新される。
② 任意のファイルを選択して Click すると、選択されたファイル名が TextBox コントロールに表示される。
③ 任意のファイルを選択して DoubleClick すると、前述②の操作に加えて、オープンするファイルと決定する。
※ 決定されたファイルのパス及びファイル名を確認するために、MsgBox 関数で表示する。

(4) TextBox コントロールの操作は、次のとおりである。

① TextBox コントロールに表示されたファイル名を編集できる。

② 任意のパスやファイル名を、TextBox コントロールに直接記述することができる。

(a) パスやファイル名を""で囲んで記述した場合は、記述されたとおりのパス及びファイルとして取扱う。⇨ 例) "D:¥Home¥User1XX¥VB¥Reidai5¥Rei-5-1.TXT"

(b) パスやファイル名を""で囲まずに記述した場合は、カレントディレクトリをパスの起点としてその配下のパス及びファイルとして取扱う。

⇨ 例) Reidai5¥Rei-5-1.TXT

③ Enter を押下すると、表示されたファイルをオープンするファイルと決定する。

※ 決定されたファイルのパス及びファイル名を確認するために、MsgBox 関数で表示する。

(5) CommandButton コントロール(開 く)を Click すると、TextBox コントロールに表示されたファイルをオープンするファイルと決定する。

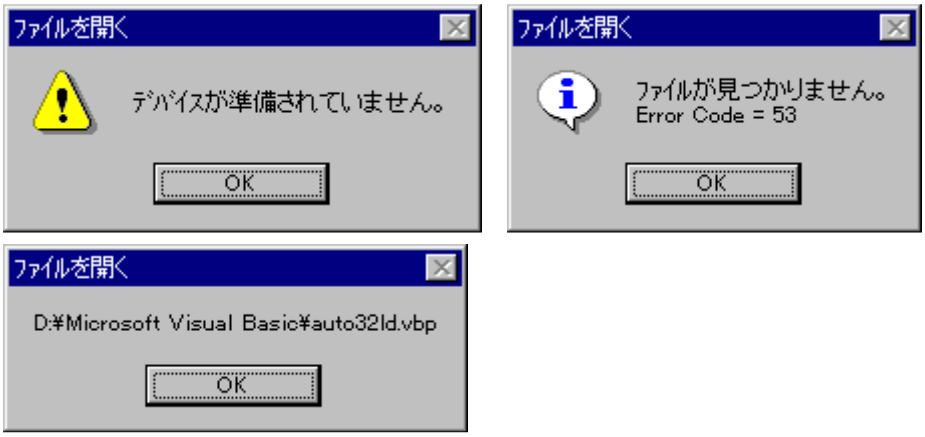
※ 決定されたファイルのパス及びファイル名を確認するために、MsgBox 関数で表示する。

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

(7) 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。

(8) 決定されたファイルのパス及びファイル名を確認する時の、実行時のエラー処理を行う。

〔MsgBox 関数の表示例〕



【実装】

(1) **例題 5-2** 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) TextBox コントロールを、Form1 フォームに追加配置する。

(3) TextBox コントロール内で、Enter が押下されたことを検出するには、KeyPress **イベント**を使用する。

【構文】

*object*_KeyPress(KeyAscii As Integer)

object … オブジェクト名(コントロール名)

KeyAscii … ANSI 文字コードを受け取る。

このプロシージャでは、TextBox コントロールに有効な文字列が格納されていて、且つ Enter が押下された場合、決定されたパス及びファイル名を確認する処理を呼び出す。

(4) 文字列の中から、指定した文字列を検索する文字列処理関数は、次のとおりである。

【構文】

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

start … 検索の開始位置を表す数式を指定。
string1… 検索対象となる文字列式を指定。
string2… 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。
戻り値 … 最初に見つかった文字位置。

【構文】

Len(*string* | *varname*)

string … 任意の文字列式を指定。
varname… 任意の変数を指定。
戻り値 … 文字列の文字数、又はバイト数
解説：Len()関数は、指定した文字列の文字数、又は指定した変数に必要なバイト数を表す長整数型(Long)の値を返す。

(5) 決定されたファイルのパス及びファイル名の有効性を確認するために、Dir()関数を使用する。

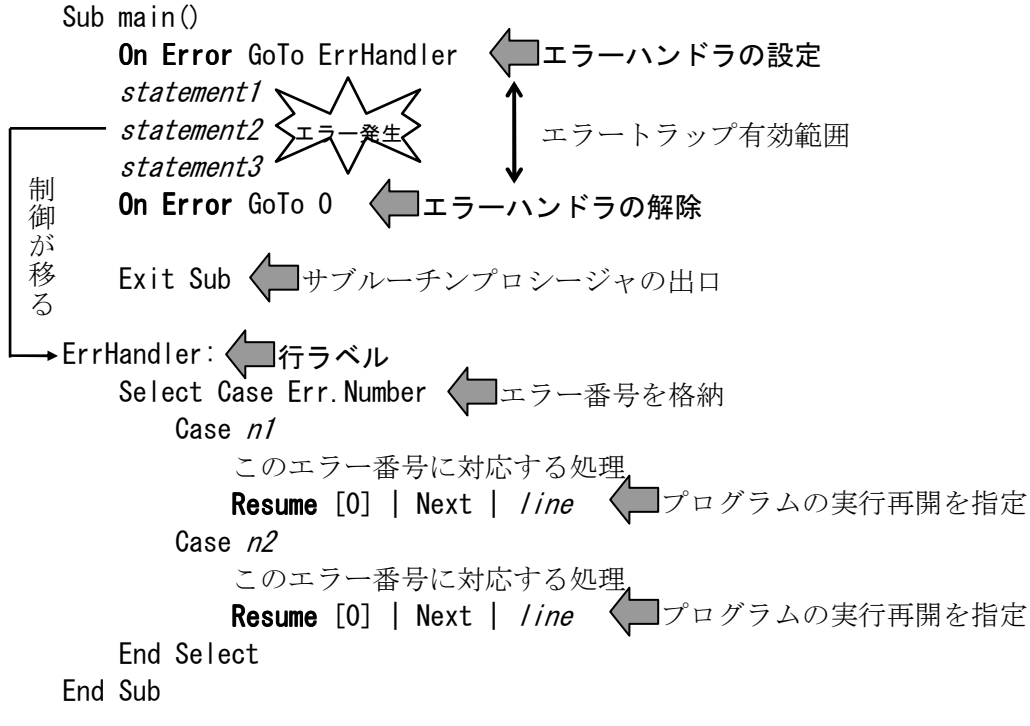
【構文】

Dir[(*pathname*[, *attributes*])]

pathname … ファイル名を表す文字列式を指定。ドライブ名及びパスも指定可能。
attributes … ファイルが持つ属性の値の合計を表す数式、又は定数を指定。
戻り値 … パス又はファイル名を表す文字列。
解説：指定したパターンやファイル属性と一致するファイル、又はフォルダの名前を表す文字列型(String)の値を返す。指定したファイル、又はフォルダが見つからない時は、長さが 0 の文字列を返す。

エラー処理(エラートラッピング)

エラーハンドラの概要を次に示す。



応用 5-3 応用 5-1 と応用 5-2 を利用して、任意の順編成ファイルを読み込み、画面に表示するアプリケーションを作成しなさい。

D:¥Home¥User1XX¥VB¥Reidai5

Project1(ReadFile.vbp)

frmSplash(Splash.frm) … Form1

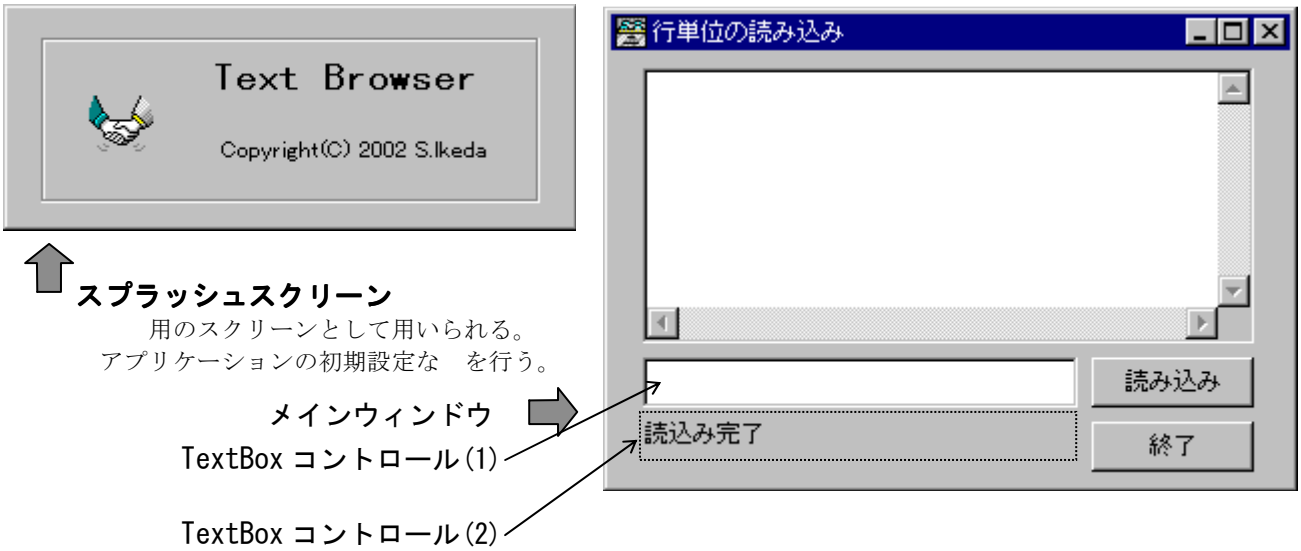
frmBrowser(ReadFile.frm) … Form2

frmOpenDLG(OpenDLG.frm) … Form3

プロジェクトの設定項目

スタートアップの設定	frmSplash … 1
タイトル	行単位の読み込み(応用 5-3)
アイコン	CRDFLE12.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
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 を行う。

メインウィンドウ

TextBox コントロール(1)

TextBox コントロール(2)

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¥Apply5-3.EXE

【機能】

(1) アプリケーションが起動された時、 中 に**スプラッシュスクリーン(Splash screen)**を 5 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 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 表示後、 中 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(読み込み)を Click すると、**開くﾀﾞｲｱﾛｸﾞ**を 中 に表示する(**応用 5-2** で作成したアプリケーション)。

(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、読み込んでいるファイルのパスとファイ

ル名を表示する。

(8) 順編成ファイルの読み込みが　了した時、**メインウィンドウ**の TextBox コントロール(2)に、**"読込み完了"**と表示する。

( ) **開くﾀﾞｲｱﾛｸﾞ**は、**タスクバー**に　　しない。

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

(11) **開くﾀﾞｲｱﾛｸﾞ**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。

(12) **開くﾀﾞｲｱﾛｸﾞ**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。

(13) **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。

【実装】

(1) **応用 5-1** と**応用 5-2** 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) **スプラッシュスクリーン**の実装は、次のとおりである。

① frmSplash フォームの**キャプションバー**を　表示にする。

- BoderStyle プロパティ　　: 3 - **固定ﾀﾞｲｱﾛｸﾞ**　　… リサイズ不可
- Caption プロパティ　　:
- ControlBox プロパティ　　: False
- ShowInTaskbar プロパティ　: False　　… タスクバー

② Frame コントロールを、frmSplash フォームに配置する。

③ Label コントロール(Text Browser)を、frmSplash フォームに配置する。

- Font プロパティ　　: **MS ゴシック**　　Bold　14

Label コントロール(Copyright(c))を、frmSplash フォームに配置する。

- Font プロパティ　　: **MS P ゴシック**　　Regular 14

Image コントロールを、frmSplash フォームに配置する。

- Picture プロパティ　　: ICONS¥COMM¥HANDSHAK.ICO

Timer コントロールを、frmSplash フォームに配置する。

- Interval プロパティ: 5000

フォームを　　中　に表示する操作には、Screen オブジェクトを使用する。

Timer コントロールの起動・停止には、Enabled プロパティを操作する。

Timer コントロールがタイムアップした時の処理は、Timer1_Timer イベントプロシージャに記述する。

frmBrowser フォームをメモリに読み込むには、Load ステートメントを使用する。

ただし、このフォームの　表示操作と frmBrowser フォームの表示操作は、このモジュールでは行わない。

(3) **メインウィンドウ**の実装は、次のとおりに変更する。

① TextBox コントロール(1)を、frmBrowser フォームに追加配置する。

② TextBox コントロール(2)を、frmBrowser フォームに追加配置する。

- Appearance プロパティ　　: 0 - ﾌﾗｯﾄ
- BackColor プロパティ　　: &H80000004&　　(Form)と同　させる

・ BoderStyle プロパティ　　：0 - なし
・ Icon プロパティ　　：ICONS¥OFFICE¥FILES06.ICO

③ 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　　中　に表示する一　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。

CommandButton コントロール(読み込み)のイベントプロシージャの処理は、次のとおりである。
・ 読み込むファイルのパス及びファイル名を取得する。ただし、このプロシージャでは直接 frmOpenDLG を呼び出さずに､後述　GetOpenFileName プロシージャを呼び出す。
・ 読み込むファイルを入力モードでオープンする。 ⇦ **重要事項**
・ ファイルから 1 行読み込み、TextBox コントロールに表示する。
・ 前述の処理を、ファイルが終了するまで繰り返す。
・ オープンしたファイルをクロースする。
・ このプロシージャでは、実行時のエラー処理を行う。**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 = "読込みエラー"      ' 動作 Status"読込みエラー"を表示 … TextBox(2)
    MsgBox Err.Description & Chr(&HA) & _
           "Error code = " & Err.Number, vbCritical, "ﾌｧｲﾙを開く"
```

読み込むファイルのパス及びファイル名の取得処理は、GetOpenFileName プロシージャで行う。プログラムコードは、次のとおりである。

```
Private Sub GetOpenFileName()
    frmOpenDLGStatus = True         ' ファイルオープンダイアログ有効

    frmOpenDLG.Show                 ' ファイルオープンダイアログの起動・表示

    Do While frmOpenDLGStatus       ' ファイルオープンダイアログの終了待機
        DoEvents                    ' Windows に制御を渡す
    Loop

    txtFileName = frmOpenDLG.FullPathName' ファイルオープンダイアログからファイル名の取得
    If txtFileName = "" Then        ' ファイル名取得失敗
        MsgBox "ﾌｧｲﾙが見つかりません。" & Chr(&HA) & _
               txtFileName, vbInformation, "ﾌｧｲﾙを開く"
    End If
End Sub
```

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
Public frmOpenDLGStatus As Boolean
```

(4) **開くﾀﾞｲｱﾛｸﾞ**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　　中　に表示する一　の処理を行う。
・ このフォームを表示する。

② frmOpenDLG フォームが　じられているかを確認する処理を記述する。
Form が　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。

※ Form に関するイベントプロシージャを記述する場合、コントロールな　と　なりオブジェクト名がイベントプロシージャに使用されないので　意する。

③ CommandButton コントロール(開 く)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。

・ オープンするファイルのパス及びファイル名の決定後、このフォームをメモリからアンロードする。

CommandButton コントロール(終了)を Click した時のイベントプロシージャを、次のとおりに変更する。

・ このフォームをメモリからアンロードする。

【考察】

次のステートメント、イベント及びメソッドを調査しなさい。

Load《ステートメント/イベント》

Unload《ステートメント/イベント》

QueryUnload

Terminate

Show

Hide