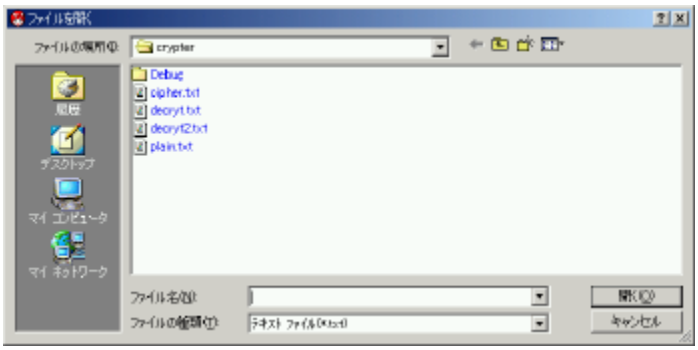
オブジェクト指向と ゲームプログラミング

● 基礎編 ― 第20回 ファイルを開く ●

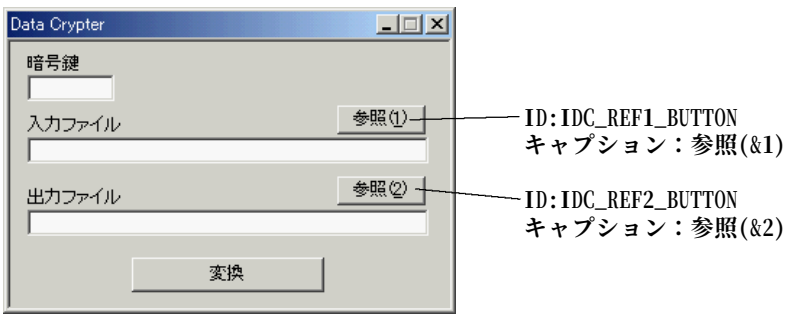
課　題

入力および出力ファイル名を「ファイルを開く」および「ファイルの保存」ダイアログで選択できるようにしましょう。

1　「ファイルを開く」および「ファイルの保存」ダイアログとは、以下のようなファイルを開いたり保存したりするときによく使われるダイアログボックスのことです。



2　ダイアログボックスに以下のようなボタンを追加しましょう。



追加した「参照」ボタンを押すと、「ファイルを開く」「ファイルの保存」ダイアログが表示され、ファイルが選択できるように、プログラムを作成してきます。

3　「参照(1)」ボタンを押したら、「ファイルを開く」ダイアログが表示され、入力ファイルの選択ができるようにしましょう。

「ファイルを開く」ダイアログは、GetOpenFileName関数を呼び出すだけで表示することができます。しかし、この関数の引数には、多数のメンバを持つOPENFILENAME構造体を正しく設定して渡さなければなりません。

OPENFILENAME構造体は、以下のメンバで構成され、ファイルを選択するダイアログを初期化するための情報を格納します。GetOpenFileName関数のほかに「ファイルの保存」ダイアログを呼び出すGetSaveFileName関数でも使用します。

```
struct OPENFILENAME {
  DWORD         lStructSize;        // この構造体のサイズをバイト単位で指定
  HWND          hwndOwner;          // 親ウィンドウのハンドルを指定
  HINSTANCE     hInstance;          // 通常はNULLを指定
  LPCTSTR       lpstrFilter;        // ファイル選択のフィルタ文字列を指定
  LPTSTR        lpstrCustomFilter;  // カスタムフィルタの指定。通常はNULL
  DWORD         nMaxCustFilter;     // カスタムフィルタのバッファサイズ
```

```
    DWORD          nFilterIndex;          // 表示させるフィルタのインデックスを指定
    LPTSTR         lpstrFile;             // フルパスファイル名を格納する文字配列を指定
    DWORD          nMaxFile;              // フルパスファイル名を格納する領域のサイズを指定
    LPTSTR         lpstrFileTitle;        // ファイル名のみを格納する文字配列を指定
    DWORD          nMaxFileTitle;         // ファイル名のみを格納する領域のサイズを指定
    LPCTSTR        lpstrInitialDir;       // 初期ディレクトリの指定
    LPCTSTR        lpstrTitle;            // ダイアログのタイトルバーに表示される文字列を指定
    DWORD          Flags;                 // ダイアログの機能を示すフラグの組み合わせ
    WORD           nFileOffset;           // ファイル名までのオフセット
    WORD           nFileExtension;        // ファイルの拡張子までのオフセット
    LPCTSTR        lpstrDefExt;           // デフォルト拡張子の文字列を指定
    LPARAM         lCustData;             // フックプロシージャへの引数を指定
    LPOFNHOOKPROC lpfnHook;               // フックプロシージャの指定。通常はNULLを指定
};
```

以上をまとめると、以下のようなプログラムになります。これを参考に、「参照(1)」ボタンを押したときの処理を追加しましょう。

```
    TCHAR          szFileName[MAX_PATH + 1] = "";    // ファイル名
    OPENFILENAME   ofn;                              // ダイアログ属性設定

    // OPENFILENAME構造体の設定
    ofn.lStructSize       = sizeof(ofn);
    ofn.hwndOwner         = hDlg;
    ofn.hInstance         = NULL;
    ofn.lpstrFilter       = "ﾃｷｽﾄ ﾌｧｲﾙ(*.txt)\0*.txt\0すべてのﾌｧｲﾙ(*.*)\0*.*\0\0";
    ofn.lpstrCustomFilter = NULL;
    ofn.nMaxCustFilter    = NULL;
    ofn.nFilterIndex      = 0;
    ofn.lpstrFile         = szFileName;
    ofn.nMaxFile          = sizeof(szFileName);
    ofn.lpstrFileTitle    = NULL;
    ofn.nMaxFileTitle     = 0;
    ofn.lpstrInitialDir   = NULL;
    ofn.lpstrTitle        = "ファイルを開く";
    ofn.Flags             = OFN_FILEMUSTEXIST | OFN_HIDEREADONLY;
    ofn.nFileOffset       = 0;
    ofn.nFileExtension    = 0;
    ofn.lpstrDefExt       = "txt";
    ofn.lCustData         = 0;
    ofn.lpfnHook          = NULL;
    ofn.lpTemplateName    = NULL;

    // 「ファイルを開く」ダイアログの呼び出し
    GetOpenFileName(&ofn);
```

4　「参照(2)」ボタンを押したら、「ファイルの保存」ダイアログが表示され、出力ファイルの設定ができるようにしましょう。

5　「ファイルを開く」および「ファイルの保存」ダイアログで設定されたファイル名をエディットに表示されるようにしましょう。

応用問題

ダイアログボックスプロシージャの処理をメッセージごとに関数化すると、プログラムが管理しやすくなります。
以下のプログラムは、ダイアログボックスプロシージャを関数化したものです。足りない部分を補って完成させましょう。

```
/*
================================================================================================
                                        ツールプログラミング
    Programmed by Hibikino software. Copyright (c) 2004 Hibikino software. All rights reserved.
================================================================================================

  【対象ＯＳ】
    Microsoft Windows98/2000以降

  【コンパイラ】
    Microsoft VisualC++ 6.0J ServicePack6

  【プログラム】
    crypter.cpp
                データ暗号化・復号

  【履歴】
    * Version    1.00       2004/10/dd  hh:mm:ss

================================================================================================
*/

/************************************************************************************************/
/*                                        インクルードファイル                                  */
/************************************************************************************************/
#include <windows.h>
#include <stdio.h>
#include "resource.h"

/************************************************************************************************/
/*                                          プロトタイプ                                        */
/************************************************************************************************/
BOOL CALLBACK DialogProc(HWND hDlg, UINT uMsg, WPARAM wParam, LPARAM lParam);

BOOL OnCommand   (const HWND hDlg, const WPARAM wParam, const LPARAM lParam);
BOOL OnInitDialog(const HWND hDlg, const WPARAM wParam, const LPARAM lParam);
BOOL OnClose     (const HWND hDlg, const WPARAM wParam, const LPARAM lParam);

bool Convert          (const HWND hDlg);
bool GetInputFileName (const HWND hDlg);
bool GetOutputFileName(const HWND hDlg);

/************************************************************************************************/
/*                                         WinMain関数                                          */
/************************************************************************************************/
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpszCmdLine, int nShowCmd)
{
    return DialogBox(hInstance, MAKEINTRESOURCE(IDD_DIALOG), NULL, DialogProc);
}

/************************************************************************************************/
/*                                     ダイアログボックスプロシージャ                           */
/************************************************************************************************/
BOOL CALLBACK DialogProc(HWND hDlg, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    switch(uMsg) {
      case WM_COMMAND:      return ※ここは各自考えましょう(hDlg, wParam, lParam);
      case WM_INITDIALOG:   return ※ここは各自考えましょう(hDlg, wParam, lParam);
      case WM_CLOSE:        return ※ここは各自考えましょう(hDlg, wParam, lParam);
    }

    return 0;
}

/************************************************************************************************/
/*                                    WM_COMMANDメッセージ処理                                  */
```

```
/*****************************************************************************************************/
BOOL OnCommand(HWND hDlg, WPARAM wParam, LPARAM lParam)
{
    switch(LOWORD(wParam)) {
      case IDC_CONV_BUTTON:     ※ここは各自考えましょう;   return 1;   // 変換ボタン
      case IDC_REF1_BUTTON:     ※ここは各自考えましょう;   return 1;   // 参照(入力)ボタン
      case IDC_REF2_BUTTON:     ※ここは各自考えましょう;   return 1;   // 参照(出力)ボタン
    }

    return 0;
}


/*****************************************************************************************************/
/*                                          データ変換                                               */
/*****************************************************************************************************/
bool Convert(const HWND hDlg)
{
    ※この関数の内容は各自考えましょう
}

/*****************************************************************************************************/
/*                                        入力ファイル名取得                                         */
/*****************************************************************************************************/
bool GetInputFileName(const HWND hDlg)
{
    ※この関数の内容は各自考えましょう
}

/*****************************************************************************************************/
/*                                        出力ファイル名取得                                         */
/*****************************************************************************************************/
bool GetOutputFileName(const HWND hDlg)
{
    ※この関数の内容は各自考えましょう
}

/*****************************************************************************************************/
/*                                    WM_INITDIALOGメッセージ処理                                    */
/*****************************************************************************************************/
BOOL OnInitDialog(HWND hDlg, WPARAM wParam, LPARAM lParam)
{
    ※この関数の内容は各自考えましょう
}

/*****************************************************************************************************/
/*                                     WM_CLOSEメッセージ処理                                        */
/*****************************************************************************************************/
BOOL OnClose(HWND hDlg, WPARAM wParam, LPARAM lParam)
{
    ※この関数の内容は各自考えましょう
}
```