

XÂY DỰNG ỨNG DỤNG DIALOG-BASED

Văn Chí Nam – Nguyễn Đức Hoàng Hạ

Khoa Công nghệ Thông tin, Trường ĐH KHTN TP.HCM

(vcnam@fit.hcmuns.edu.vn, ndhha@fit.hcmuns.edu.vn)

Phiên bản cập nhật ngày 12/10/2004

MỤC ĐÍCH

Bài viết này giúp cho người đọc làm quen và có thể thực hiện được các thao tác cơ bản trên dialog. Bài viết này cũng cung cấp những tham khảo cơ bản dành cho một số lớp đối tượng quen thuộc trên MFC.

GIỚI THIỆU SƠ NÉT

Trong phần giới thiệu này, chúng ta sẽ làm quen với 1 vài lớp đối tượng cơ bản, cơ sở trên MFC.

Lớp CWnd

Đây là một lớp tổng quát và gặp rất nhiều lần trong quá trình làm việc trên dialog nói riêng và trên các ứng dụng MFC nói chung.

Lớp CWnd cung cấp các chức năng cơ bản cho tất cả các lớp cửa sổ¹ (các control, mainframe, view, dialog...) trong thư viện MFC.

EnableWindow

```
BOOL EnableWindow( BOOL bEnable = TRUE );
```

GetFocus

```
static CWnd* PASCAL GetFocus( );
```

Trả về con trỏ CWnd của control đang được nhận focus.

IsEnableWindow

```
BOOL IsWindowEnabled( ) const;
```

SetFocus

Đặt focus cho một đối tượng cửa sổ.

```
CWnd* SetFocus( );
```

¹ Từ vị trí này, thuật ngữ *cửa sổ* được hiểu chung cho các đối tượng thuộc nhóm này : dialog, các control, view, mainframe...

GetClientRect

Trả về tọa độ của hình chữ nhật bao quanh cửa sổ.

```
void GetClientRect( LPRECT lpRect ) const;
```

Ví dụ :

```
CRect rect;  
pWnd->GetClientRect(&rect);
```

MoveWindow

Dịch chuyển, thay đổi kích cỡ một cửa sổ

```
void MoveWindow(int x, int y, int nWidth, int nHeight,  
BOOL bRepaint = TRUE );
```

```
void MoveWindow(LPCRECT lpRect, BOOL bRepaint = TRUE);
```

Ví dụ :

```
this->MoveWindow(100, 100, 400, 400);
```

```
CRect rect;  
rect.top = 100;  
rect.left = 100;  
rect.right = 500;  
rect.bottom = 500;  
this->MoveWindow(&rect);
```

GetDlgItem

Trả về con trỏ CWnd* của một cửa sổ con trên 1 dialog.

```
CWnd* GetDlgItem(int nID ) const;
```

Ví dụ :

```
CWnd *pWnd;  
pWnd = GetDlgItem(IDC_EDIT1);
```

GetDlgItemID

Trả về ID của một cửa sổ trên dialog.

```
int GetDlgItemID() const;
```

GetDlgItemInt

```
UINT GetDlgItemInt(int nID, BOOL* lpTrans = NULL,  
BOOL bSigned = TRUE ) const;
```

Trả về giá trị số nguyên của 1 control trên Dialog. (Dùng trong trường hợp không muốn nhận giá trị chuỗi của 1 số nguyên rồi chuyển từ chuỗi sang số).

GetDlgItemText

```
int GetDlgItemText(int nID, LPTSTR lpStr, int nMaxCount) const;
```

```
int GetDlgItemText(int nID, CString& rString) const;
```

Lấy chuỗi nội dung của 1 control.

Ví dụ : Lấy chuỗi nội dung của control có ID là IDC_EDIT1 trên dialog.

```
CString str;
GetDlgItem(IDC_EDIT1, str);
// str sẽ mang nội dung của IDC_EDIT1
```

SetDlgItemText

```
void SetDlgItemText(int nID, LPCTSTR lpszString);
```

Đặt chuỗi nội dung cho một control trên dialog.

GetWindowText

Lấy nội dung cho một đối tượng cửa sổ

Ví dụ : Lấy nội dung cho đối tượng cửa sổ có ID là IDC_TEXT

```
CWnd * pWnd;
CString str;
pWnd = GetDlgItem(IDC_TEXT);
pWnd->GetWindowText(str);
```

SetWindowText

Đặt nội dung cho một đối tượng cửa sổ

Ví dụ : Đặt nội dung cho đối tượng cửa sổ có ID là IDC_TEXT

```
CWnd * pWnd;
CString str;
pWnd = GetDlgItem(IDC_TEXT);
pWnd->SetWindowText("Hi ! Hello");
```

Lớp CString

Tham khảo tài liệu “Sử dụng các cấu trúc dữ liệu đơn giản trên MFC” của thầy Lu Buon Vinh (thelastsamuraitor@yahoo.ca).

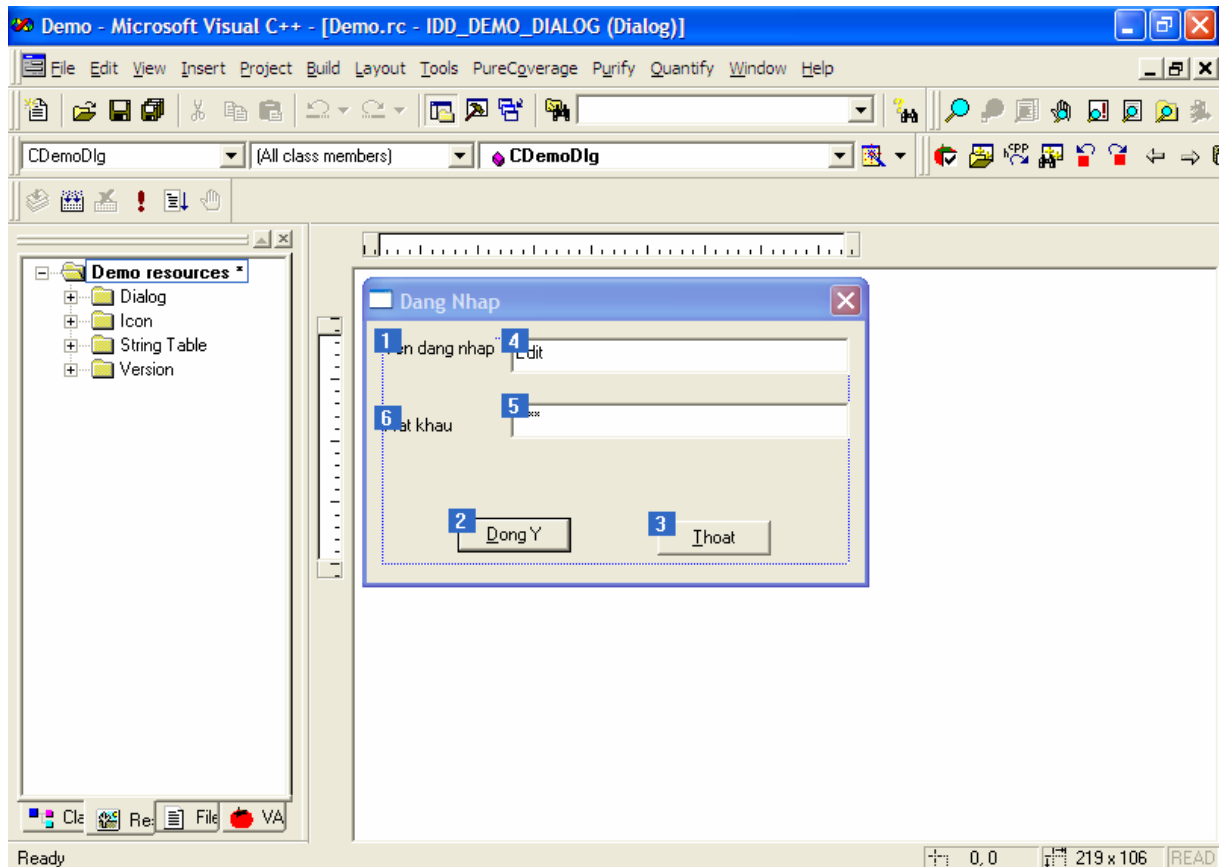
LƯU Ý TRONG THIẾT KẾ GIAO DIỆN

Thứ tự tab của các control (Tab-Order)

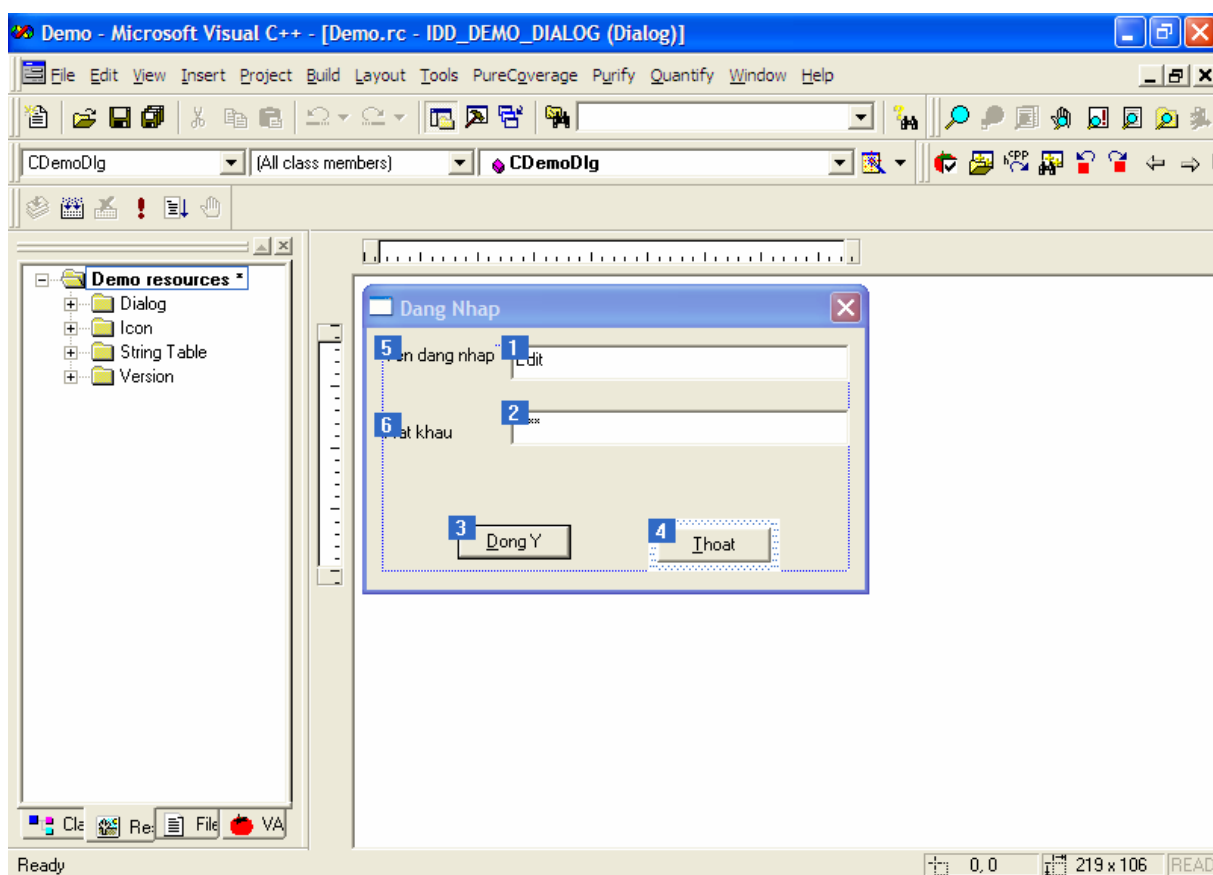
Các control trên dialog cần được tổ chức theo một thứ tự nhất định nhằm tạo tiện lợi cho người sử dụng khi dùng bàn phím. Người sử dụng sẽ sử dụng phím Tab trên bàn phím để dịch chuyển từ một control này đến một control khác trên dialog. Người thiết kế dialog cần chú ý đặc điểm này khi thiết kế. Các control kế tiếp nhau phải có thứ tự tab kế tiếp nhau.

Sử dụng phím tắt **Ctrl-D** để bật chức năng Tab Order. (Có thể vào menu Layout/Tab Order).

Lần lượt click chọn các control theo thứ tự tăng dần của thứ tự tab. (Đừng quan tâm đến các static text !).



Xây dựng ứng dụng dialog-based trên Visual C++ 6.0



Dùng chức năng hiển thị dialog lúc thiết kế (**Ctrl-T**) để kiểm tra xem thứ tự tab giữa các control đã phù hợp chưa.

Canh chỉnh control

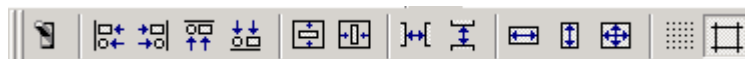
Chức năng canh chỉnh

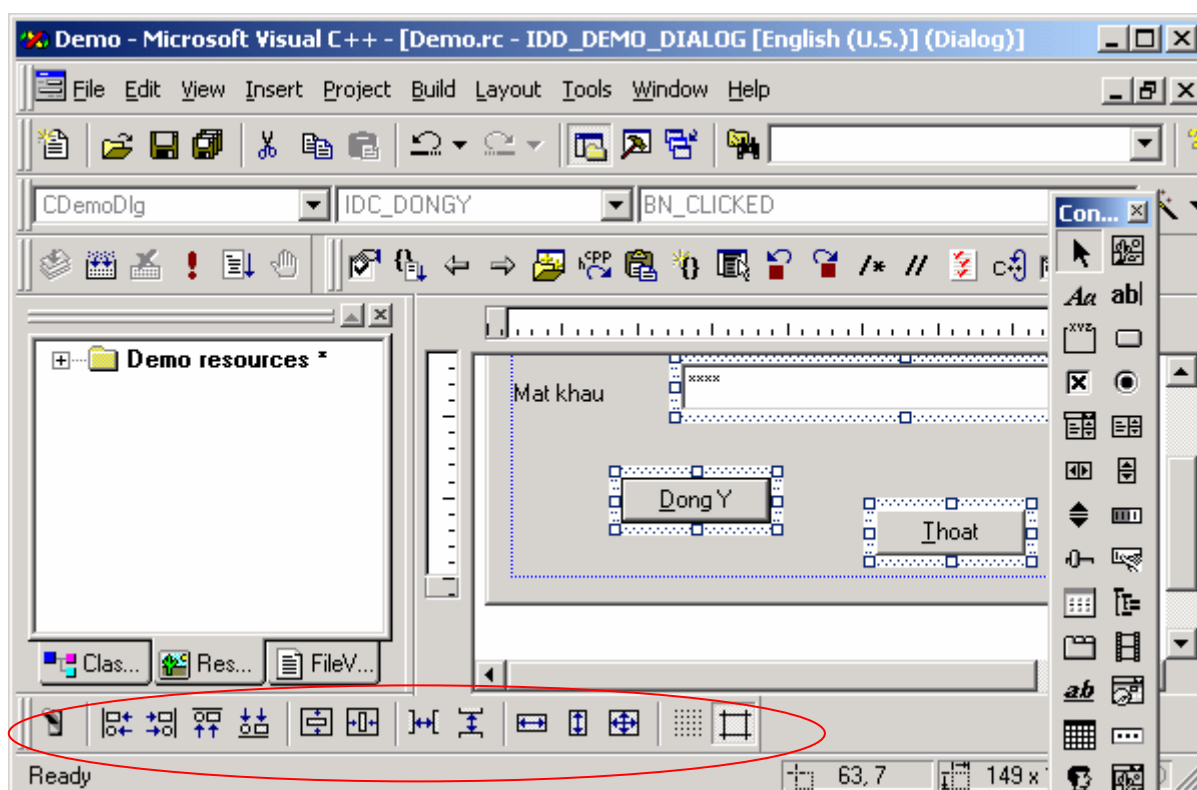
Các chức năng canh chỉnh có thể thực hiện:

- Canh chỉnh 2 hay nhiều control cùng kích cỡ (chiều dài, chiều rộng, hoặc cả hai chiều).
- Canh chỉnh 2 hay nhiều control theo cạnh trái / phải / trên / dưới của một control.
- Canh giữa control theo chiều cao, chiều rộng của dialog.
- Canh đều khoảng cách các control theo chiều cao, chiều rộng của dialog.

Nguyên tắc thực hiện

- Chọn lựa các control bằng cách nhấn chọn và giữ phím Ctrl hoặc Shift.
- Control được chọn làm chuẩn sẽ được chọn sau cùng.
- Chọn chức năng cần thực hiện trên thanh công cụ sau :





LÀM QUEN VỚI BIẾN ĐẠI DIỆN

Biến đại diện

Biến đại diện giúp lập trình viên điều khiển, xác định, cập nhật giá trị của một control một cách nhanh chóng, tiện lợi. Có 2 loại biến đại diện : biến đại diện *kiểu control* và biến đại diện *kiểu dữ liệu*. Với biến đại diện kiểu control, chúng ta có thể thao tác trên biến như thao tác trên đối tượng trực tiếp (có nghĩa là có thể dùng mọi phương thức đi kèm với lớp đối tượng). Với biến đại diện kiểu dữ liệu, chúng ta chỉ có thể xác định, cập nhật giá trị thể hiện trên nó. (Cùng 1 control nhưng có thể có các kiểu dữ liệu khác nhau, ví dụ, trên Edit Box, chúng ta có thể tạo biến đại diện kiểu *CString*, kiểu *int*, kiểu *UINT*... theo yêu cầu của bài toán.).

Xác định, cập nhật giá trị biến đại diện

Sử dụng hàm **UpdateData** để xác định hay cập nhật giá trị của một biến đại diện.

Với **UpdateData(TRUE)**, chúng ta có thể đưa giá trị hiện hành của control vào trong biến đại diện.

Với **UpdateData(FALSE)**, chúng ta sẽ cập nhật giá trị control bằng giá trị hiện hành của biến đại diện.

Các bước tạo biến đại diện

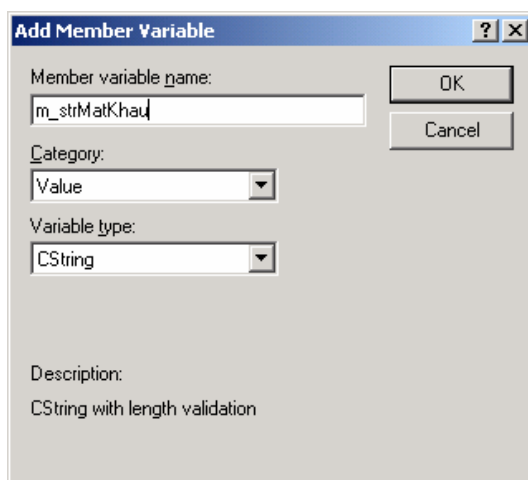
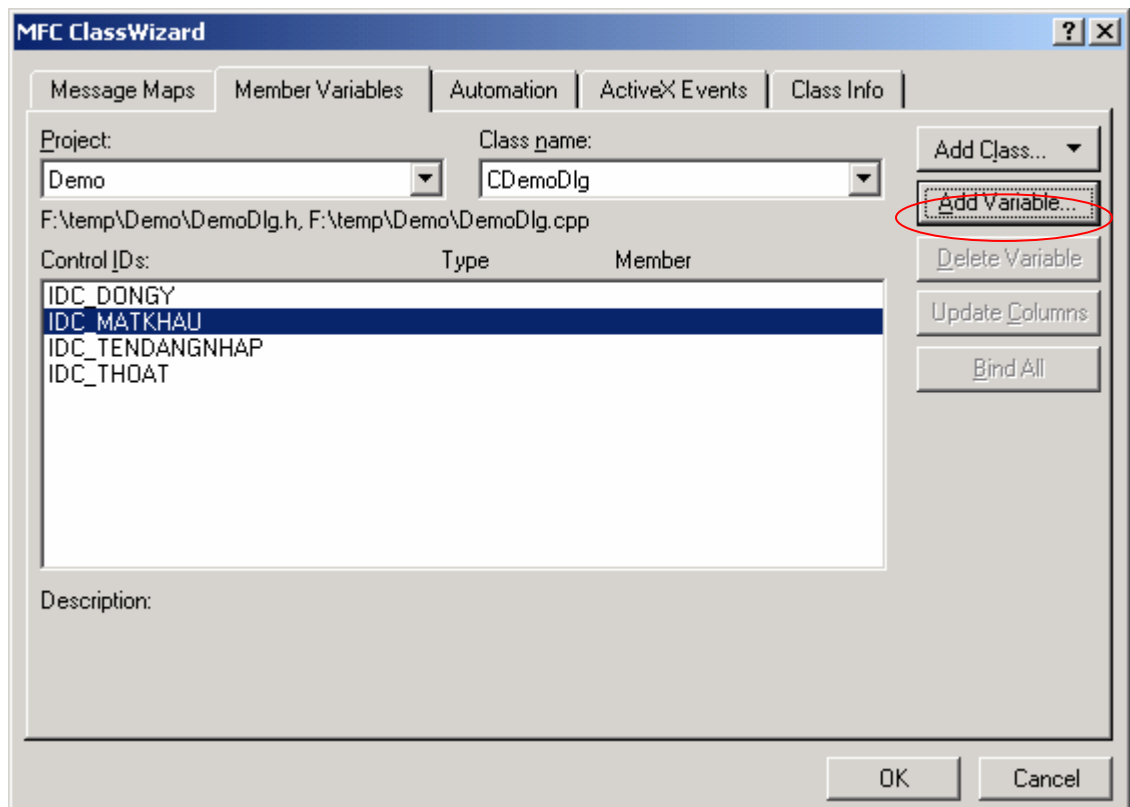
Bước 1 : Sử dụng tab **Member Variables** trong **ClassWizard** để tạo biến đại diện cho một control.

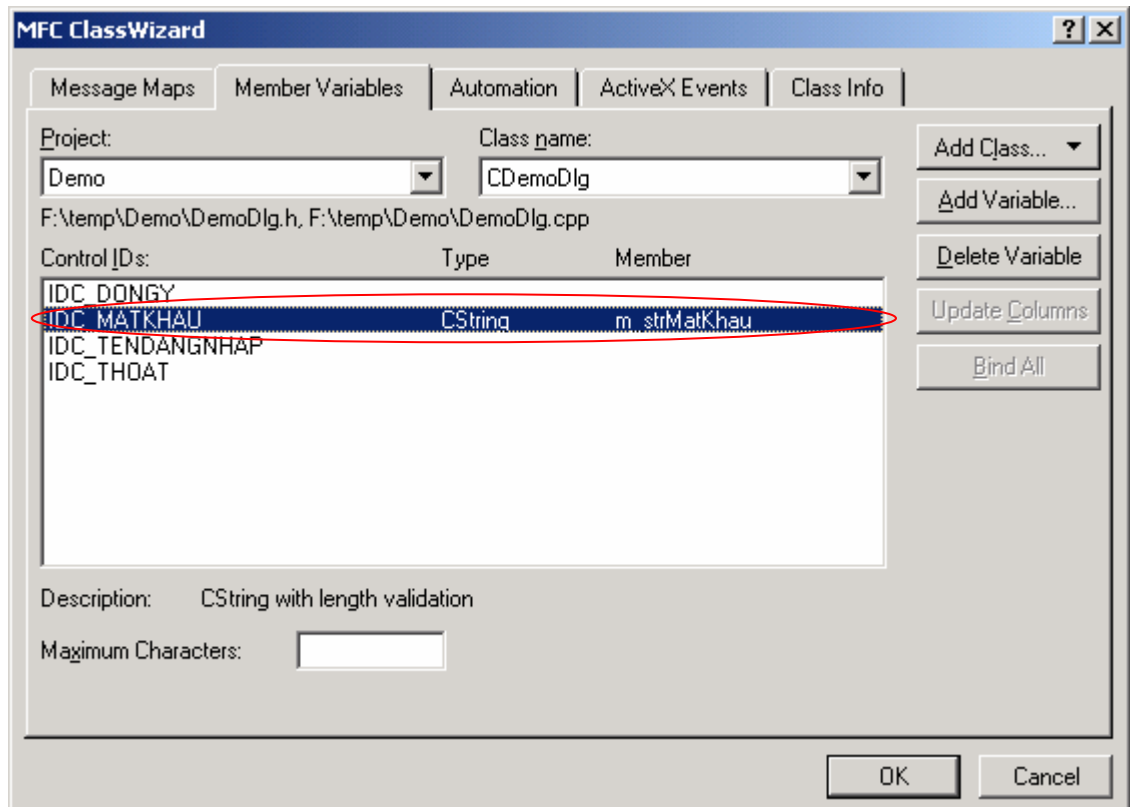
Bước 2 : Chọn ID của control cần đặt biến đại diện trong danh sách **Control IDs**.

Bước 3 : Nhấn vào **Add Variable** để bắt đầu tạo.

Bước 4 : Đặt tên biến trong ô **Member variable name**, chọn loại biến trong danh sách **Category**, chọn kiểu dữ liệu của biến trong danh sách **Variable type**.

Bước 5 : Nhấn OK để đồng ý. Nhìn lại kết quả thực hiện được.





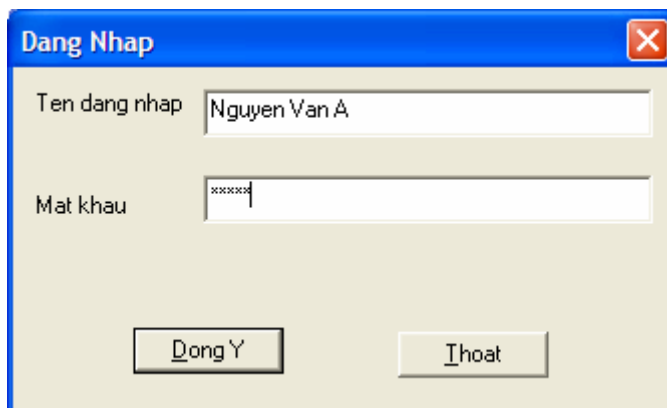
VÍ DỤ MINH HOẠ

Mô tả

Ứng dụng đơn giản mà chúng ta sẽ xây dựng trong bài viết này là một màn hình đăng nhập trong một chương trình.

Người sử dụng sẽ nhập vào tên đăng nhập và mật khẩu vào trong 2 hộp thoại tương ứng. Sau khi nhập xong, có thể nhấn vào nút **Dong Y** để kiểm tra mật khẩu. Nếu mật khẩu và tên đăng nhập hợp lệ, người sử dụng nhận được một thông báo chúc mừng. Ngược lại, người sử dụng phải tiếp tục nhập lại mật khẩu, tên đăng nhập.

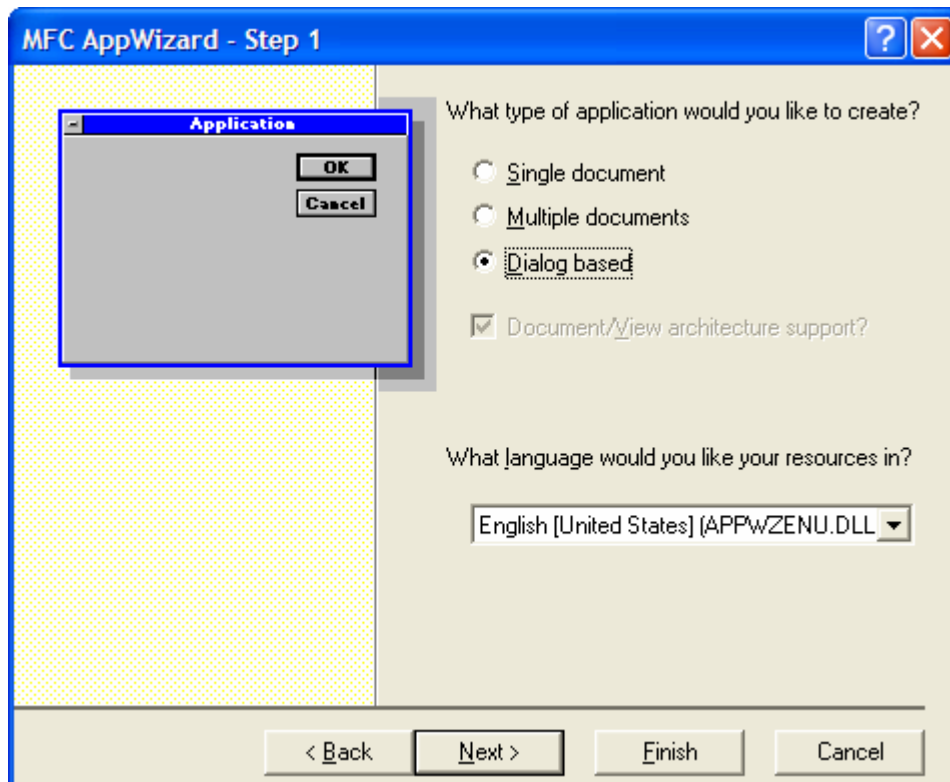
Màn hình đăng nhập được thể hiện dưới đây :



Các bước tạo project mới

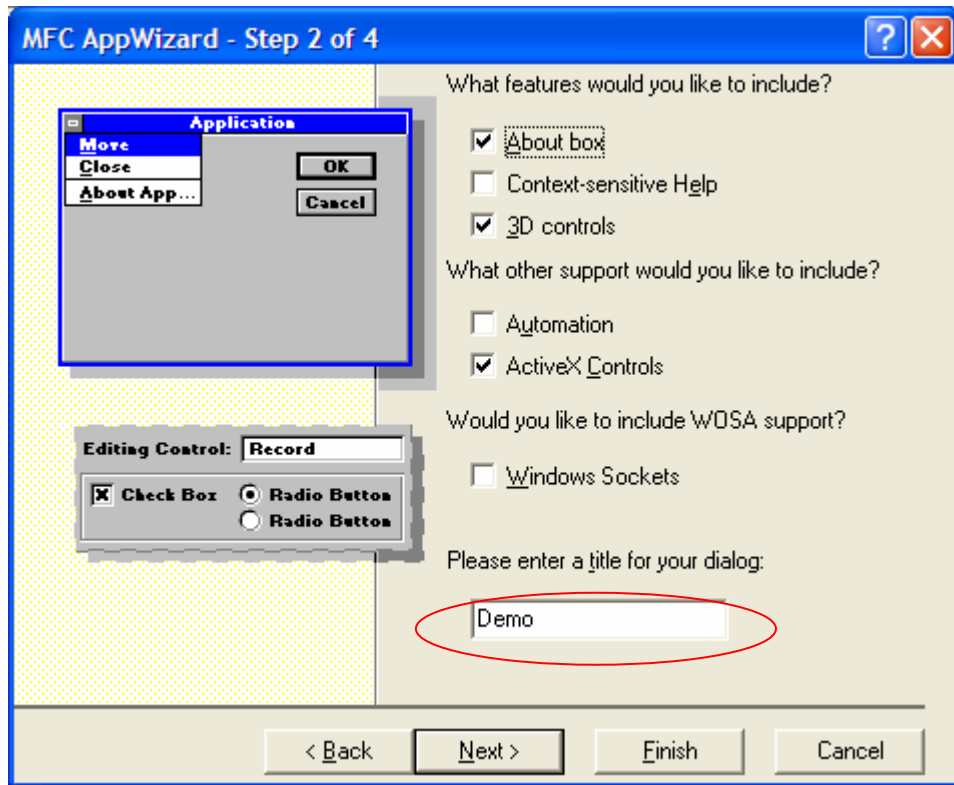
Bước 1 : Tạo một project mới sử dụng MFC AppWizard. Đặt tên cho project mới là Demo.

Bước 2 : Chọn loại ứng dụng là **Dialog-based**.

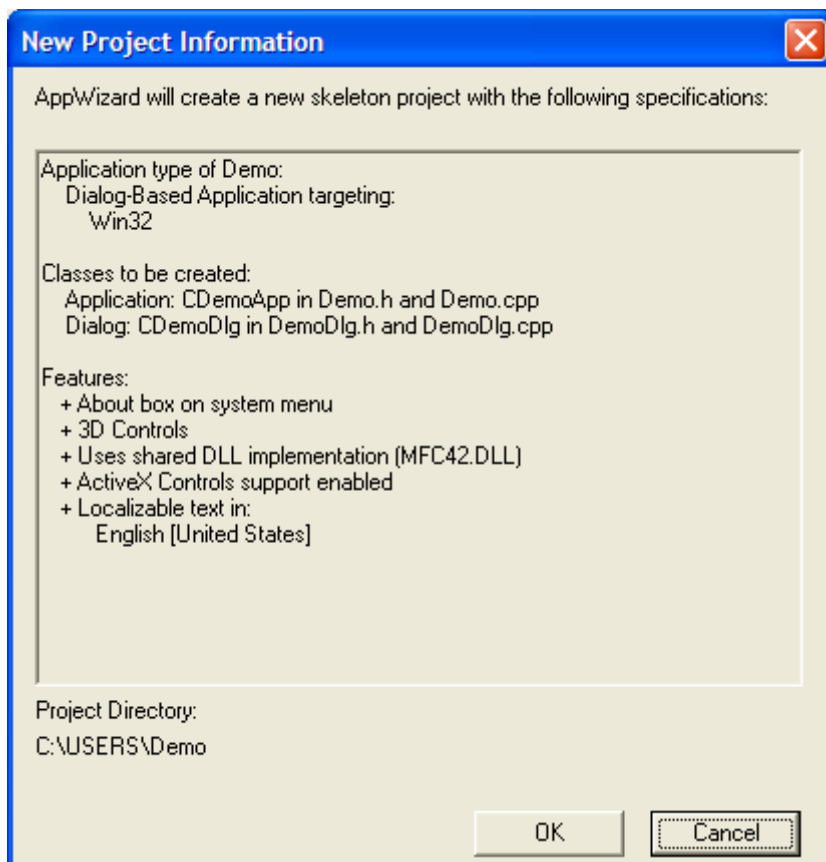


Bước 3 : Giữ nguyên các giá trị mặc định. (Lưu ý : có thể thay đổi nội dung của thanh tiêu đề (title bar) ở **Step 2**).

Xây dựng ứng dụng dialog-based trên Visual C++ 6.0



Bước 4 : Có thể nhấn **Finish** để kết thúc. Các thông tin tổng hợp về project được hiển thị ở dialog cuối cùng.



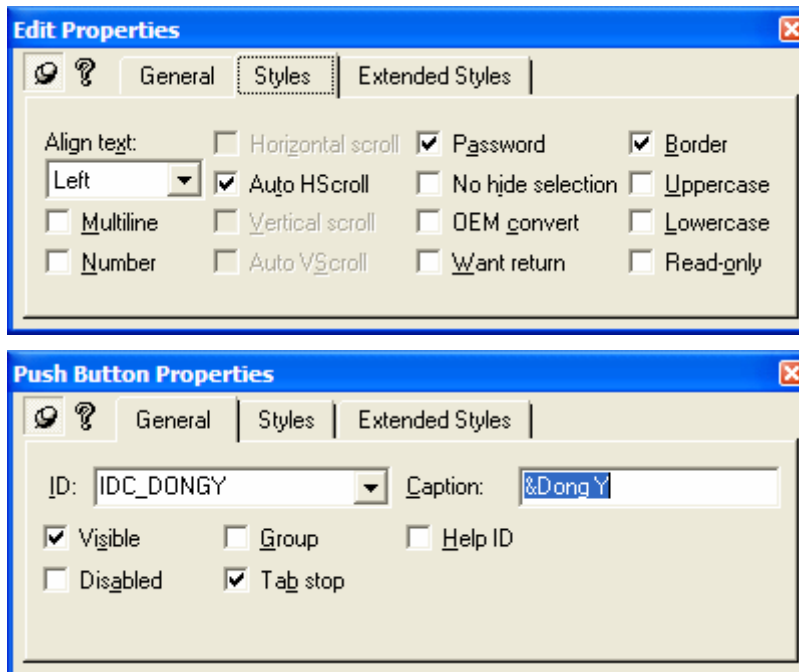
Các bước thiết kế giao diện

Chọn tab **ResourceView** phần **Workspace**. Chọn dialog mong muốn thiết kế trên nhánh **Dialog**.

Kéo thả các control đến các vị trí mong muốn trên dialog. Dialog **Đăng nhập** như mô tả phía trên cần dùng 2 *static text*, 2 *edit box*, và 2 *button*.

Đặt tên (ID) một số control theo quy ước sau :

STT	Control	Tên	Ghi chú
1	Edit box 1	IDC_TENDANGNHAP	
2	Edit box 2	IDC_MATKHAU	Chọn style Password cho control này
3	Button 1	IDC_DONGY	
4	Button 2	IDC_THOAT	

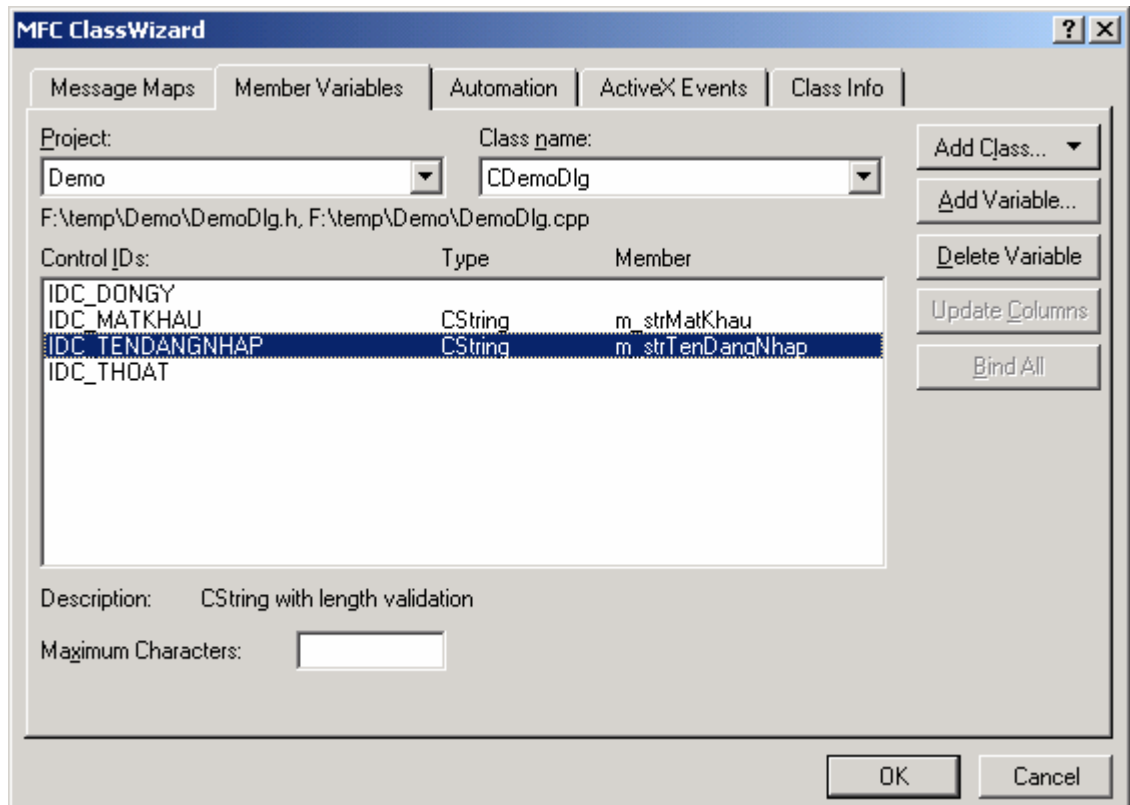


Tạo biến đại diện

Chúng ta cần xác định giá trị của 2 hộp Edit Box *Tên đăng nhập* và *Mật khẩu*. Do đó, chúng ta sẽ đặt 2 biến đại diện tương ứng với 2 control này.

ID	Biến	Kiểu	Ghi chú
IDC_MATKHAU	m_strMatKhai	CString	Mật khẩu
IDC_TENDANGNHAP	m_strTenDangNhap	CString	Tên đăng nhập

Xây dựng ứng dụng dialog-based trên Visual C++ 6.0



Viết code

Nhấn vào nút Dong Y

Xử lý sự kiện **BN_CLICKED** trên button **Dong Y**.

Quan sát các chỗ được Visual C++ 6.0 tự động thêm vào (lập trình viên có thể thêm bằng tay vào các vị trí này khi viết một hàm xử lý sự kiện riêng).

Tập tin DemoDlg.cpp (thêm 2 chỗ)

```
void CDemoDlg::OnDongy ()
{
    // TODO: Add your control notification handler code here
    //Viết code thêm ở chỗ này
}
```

```
BEGIN_MESSAGE_MAP(CDemoDlg, CDialog)
    //{{AFX_MSG_MAP(CDemoDlg)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDC_DONGY, OnDongy)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

Tập tin DemoDlg.h (thêm 1 chỗ)

```
// Generated message map functions
//{{AFX_MSG(CDemoDlg)
```

```
virtual BOOL OnInitDialog();
afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
afx_msg void OnPaint();
afx_msg HCURSOR OnQueryDragIcon();
afx_msg void OnDongy();
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
```

Trong phần xử lý việc nhấn vào nút Dong Y, chúng ta phải cập nhật lại giá trị cho các biến đại diện. Sau đó, chúng ta sẽ kiểm tra xem Tên đăng nhập (biến *m_strTenDangNhap*) và mật khẩu (biến *m_strMatKhau*) có hợp lệ không. Nếu không thì yêu cầu người sử dụng nhập lại.

```
UpdateData(TRUE);
if (m_strTenDangNhap != "test" || m_strMatKhau != "test")
{
    MessageBox("Ten dang nhap hoac mat khau khong hop
le", "Thong bao", MB_OK|MB_ICONEXCLAMATION);
    m_strMatKhau = "";
    UpdateData(FALSE);
    return;
}
```

Nếu muốn con trỏ tự động nhảy vào hộp Mật Khẩu nếu người dùng gõ sai mật khẩu thì ta có thể thêm vào đoạn mã sau :

```
CWnd * pWnd;
pWnd = GetDlgItem(IDC_MATKHAU);
pWnd->SetFocus();
```

Tóm lại, hàm xử lý sự kiện nhấn vào button **Dong Y** như sau :

```
void CDemoDlg::OnDongy()
{
    // TODO: Add your control notification handler code here
    UpdateData(TRUE);
    if (m_strTenDangNhap != "test" || m_strMatKhau != "test")
    {
        MessageBox("Ten dang nhap hoac mat khau khong hop
le", "Thong bao", MB_OK|MB_ICONEXCLAMATION);
        m_strMatKhau = "";
        UpdateData(FALSE);
        CWnd * pWnd;
        pWnd = GetDlgItem(IDC_MATKHAU);
        pWnd->SetFocus();
        return;
    }
    MessageBox("Chuc mung ban dang nhap thanh cong", "Thong
bao");
}
```

Nhấn vào nút Cancel

Trong hàm Cancel chúng ta làm duy nhất một thao tác kết thúc Dialog. Tuy nhiên, có thể thêm vào chức năng hỏi người dùng có muốn kết thúc hay không. Nếu thực sự muốn kết thúc thì sẽ cho phép kết thúc.

Để thoát khỏi màn hình một dialog, thêm vào một dòng mã sau :

```
CDialog::OnCancel();
```

Kiểm tra trả lời từ người dùng :

```
int Result;
Result = MessageBox("Ban co muon thoat khong ?", "Thong
bao", MB_YESNO|MB_ICONQUESTION);
if (Result == IDYES)
{
    //Thực hiện công việc ở đây
}
```

Tóm lại, hàm xử lý sự kiện nhấn vào button **Thoat** như sau :

```
void CDemoDlg::OnThoat()
{
    // TODO: Add your control notification handler code here
    int Result;
    Result = MessageBox("Ban co muon thoat khong ?", "Thong
bao", MB_YESNO|MB_ICONQUESTION);
    if (Result == IDYES)
    {
        CDialog::OnCancel();
    }
}
```