Lập trình C trên Windows

Kỹ thuật lập trình Visual C++ (MFC)

Nguyễn Tri Tuấn
Khoa CNTT – ĐH.KHTN.Tp.HCM
Email: nttuan@ fit.hcmuns.edu.vn

Nội dung

- Giới thiệu về MFC
- Chương trình MFC đầu tiên
- Xử lý Mouse và Keyboard
- Xử lý menu
- Toolbar, Statusbar
- Các Control
- Xây dựng và xử lý hộp thoại (Dialog box)
- Documents và Views: Scroll view, List view, Tree view
- SDI – Single Document Interface
- MDI - Multi Document Interface

Giới thiệu về MFC

- MFC là gì ?
- Một số tính năng của MFC qua từng version
- Các thành phần của 1 ứng dụng trong VC++
- Các màn hình giao diện chính của VC++ 6

Giới thiệu về MFC – MFC là gì ?

- Microsoft Foundation Class
- Là một thư viện các lớp (class, OOP) trong ngôn ngữ Visual C++, dùng cho việc lập trình trên Windows
- Được xây dựng trên cơ sở các hàm thư viện API của Windows
- Version 6 có khoảng 200 class
- Giúp cho người lập trình có thể xây dựng ứng dụng nhanh và ít tốn công sức hơn so với việc sử dụng đơn thuần các hàm thư viện API của Windows
- Ta vẫn có thể gọi các hàm Windows API trong MFC

Giới thiệu về MFC – MFC là gì ?…(tt)

- Trong 1 ứng dụng MFC, ta thường không gọi hàm Windows API trực tiếp, mà sẽ tạo các object từ những lớp của MFC, và gọi phương thức của object đó
- Đa số các phương thức của MFC class có cùng tên với những hàm Windows API
- MFC tạo ra một *Application Framework*, giúp:
  - Thiết lập kiến trúc của ứng dụng một cách nhất quán và khoa học
  - Che dấu đi nhiều phần chi tiết mà Windows API đòi hỏi, giúp developer "thảnh thơi" hơn

Giới thiệu về MFC - Một số tính năng của MFC

- Version 1:
  - Các lớp List, Array, String, Time, Date, File access,…
  - Các lớp giao diện cơ bản
  - MDI, OLE 1.0
- Version 2:
  - File open, save
  - Print preview, printing
  - Scrolling window, Splitter window
  - Toolbar, Statusbar
  - Truy xuất được đến các control của VB
  - Trợ giúp theo ngữ cảnh (Context-sensitive help)
  - DLL

Giới thiệu về MFC - Một số tính năng của MFC…(tt)

- Version 2.5:
  - Hỗ trợ ODBC (Open Database Connectivity), cho phép truy xuất đến các CSDL Access, FoxPro, SQL Server,…
  - OLE 2.01
- Version 3:
  - Hỗ trợ tab dialog (property sheet)
  - Docking control bar
- Version 3.1:
  - Hỗ trợ các control chuẩn của Windows 95
  - ODBC level 2 with Access Jet database engine
  - Các lớp Winsock phục vụ lập trình TCP/IP

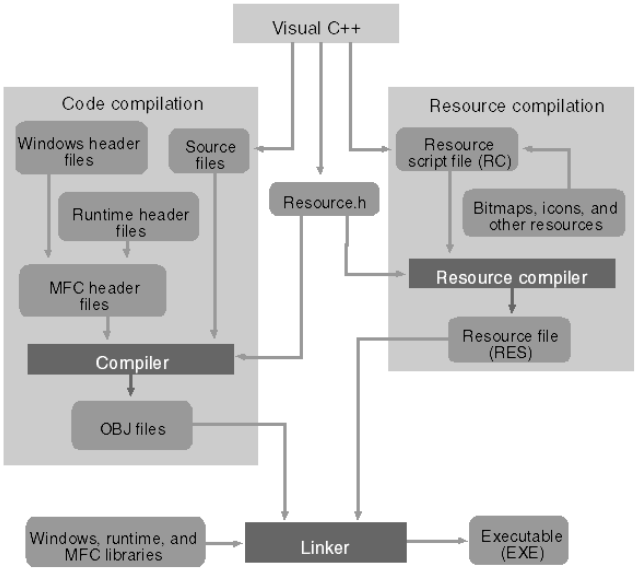
Giới thiệu về MFC - Một số tính năng của MFC…(tt)

- Version 4.0:
  - ADO (Data Access Object)
  - Windows 95 docking control bar
  - Bổ sung thêm lớp TreeView và RichEdit
  - Các lớp đồng bộ hoá các tiểu trình
- Version 4.2:
  - Các lớp WinInet
  - Các lớp ActiveX document server
  - Các tính năng mở rộng của ActiveX control
  - Tăng cường một số khả năng của ODBC

Giới thiệu về MFC - Một số tính năng của MFC…(tt)

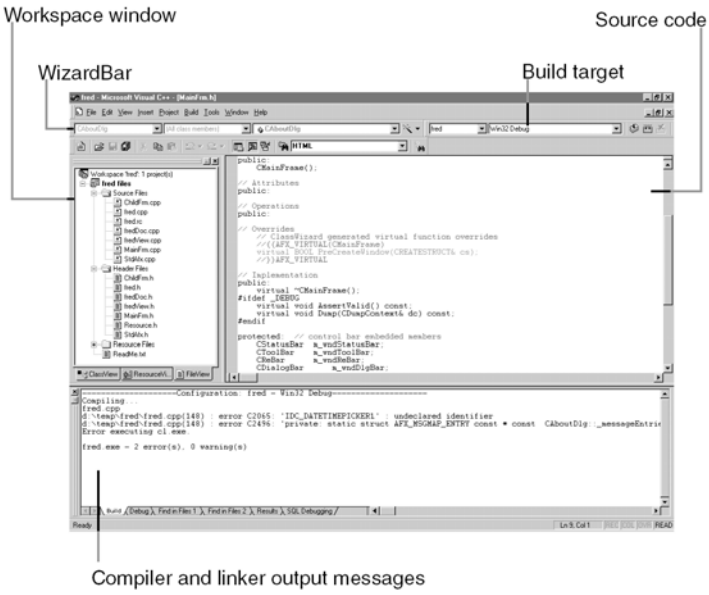
- Version 6:
  - Hỗ trợ các lớp cho những control chuẩn trong IE 4.0
  - Hỗ trợ Dynamic HTML, cho phép tạo lập động các trang HTML
  - Active Document Containment, cho phép ứng dụng MFC có thể chứa các Active Document
  - OLE DB và ADO

Giới thiệu về MFC - Các thành phần của 1 ứng dụng



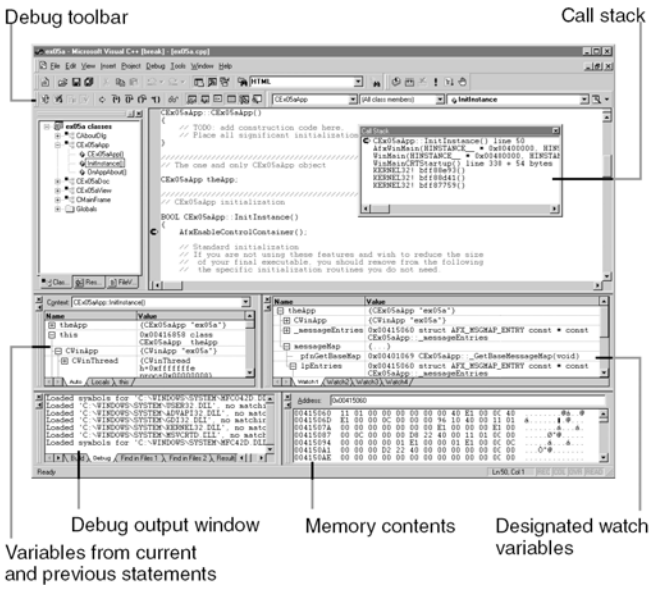
Sơ đồ biên dịch các thành phần của 1 ứng dụng trong VC++

Giới thiệu về MFC - Các màn hình giao diện chính



Các thành phần giao diện chính của VC++

Giới thiệu về MFC - Các màn hình giao diện chính…(tt)



Các thành phần giao diện chính của VC++ (run-time)

Chương trình MFC đầu tiên

- Ứng dụng đơn giản dùng Application Framework
- Ứng dụng phức tạp hơn (Dialog-based App)

Chương trình MFC đầu tiên - Ứng dụng đơn giản

- Tạo ứng dụng
- Các thành phần của chương trình

Ứng dụng đơn giản - Tạo ứng dụng

- Chọn menu File → New
- Chọn tab Projects
- Chọn loại project “Win32 Application”
- Đặt tên project và xác định đường dẫn thư mục trong ô “Location”
- Step 1: Chọn loại ứng dụng “An empty project”
- Nhấn Finish để kết thúc
- Add các file Hello.h và Hello.cpp vào project
- Chọn menu Project → Settings
  - Chọn project trong cửa sổ bên trái
  - Chọn tab General
  - Chọn “Use MFC In A Shared DLL”

Ứng dụng đơn giản - Tạo ứng dụng…(tt)

Hello.h

```
class CMyApp : public CWinApp {
        public:
                virtual BOOL InitInstance ();
};
class CMainWindow : public CFrameWnd {
        public:
                CMainWindow ();
        protected:
                afx_msg void OnPaint ();
                DECLARE_MESSAGE_MAP ()
};
```

Ứng dụng đơn giản - Tạo ứng dụng…(tt)

Hello.cpp

```
#include <afxwin.h>
#include "Hello.h"
CMyApp myApp;
//////////////////////////////////////////////////////////////////////////
// CMyApp member functions
BOOL CMyApp::InitInstance () {
        m_pMainWnd = new CMainWindow;
        m_pMainWnd->ShowWindow (m_nCmdShow);
        m_pMainWnd->UpdateWindow ();
        return TRUE;
}
//////////////////////////////////////////////////////////////////////////
// CMainWindow message map and member functions
BEGIN_MESSAGE_MAP (CMainWindow, CFrameWnd)
        ON_WM_PAINT ()
END_MESSAGE_MAP ()
```

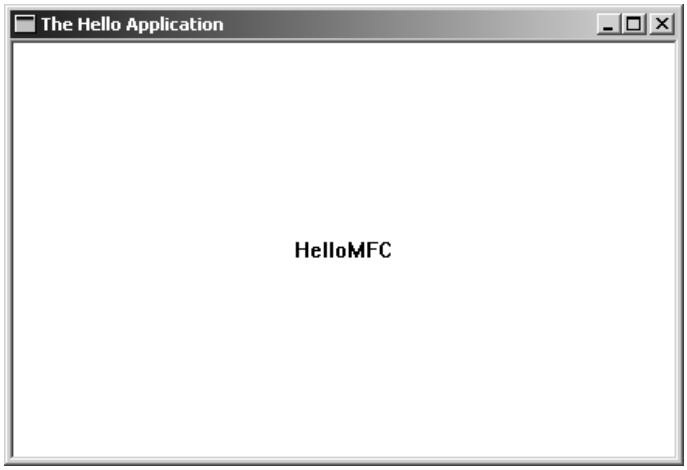

Ứng dụng đơn giản - Tạo ứng dụng…(tt)

Hello.cpp…(tt)

```
CMainWindow::CMainWindow () {
        Create (NULL, _T ("The Hello Application"));
}

void CMainWindow::OnPaint () {
        CPaintDC dc (this);
        CRect rect;
        GetClientRect (&rect);
        dc.DrawText (_T ("Hello, MFC"), -1, &rect,
                DT_SINGLELINE ¦ DT_CENTER ¦
                DT_VCENTER);
}
```

Ứng dụng đơn giản - Tạo ứng dụng…(tt)



Ứng dụng MFC đơn giản

Ứng dụng đơn giản – Các thành phần của c.trình

- CWinApp: lớp chính của MFC để quản lý ứng dụng. Chứa đựng vòng lặp nhận message và phân phối message đến các cửa sổ của ứng dụng
- CMyApp: lớp kế thừa từ lớp CWinApp

```
class CMyApp : public CWinApp {
    public:
          virtual BOOL InitInstance ();
};
```

- InitInstance(): hàm khởi tạo ứng dụng, override lên hàm chuẩn của lớp CWinApp

```
BOOL CMyApp::InitInstance () {
    m_pMainWnd = new CMainWindow;
    m_pMainWnd->ShowWindow (m_nCmdShow);
    m_pMainWnd->UpdateWindow ();
    return TRUE;
}
```

Ứng dụng đơn giản – Các thành phần của c.trình…(tt)

- CWnd: lớp chính của MFC để quản lý các loại cửa sổ giao diện. Có nhiều lớp được kế thừa từ lớp này để quản lý các loại cửa sổ khác nhau (CTreeCtrl, CListBox, Cedit,…)
- CFrameWnd: lớp kế thừa từ lớp CWnd, để quản lý cửa sổ giao diện chính của ứng dụng
- CMainWindow: lớp kế thừa từ lớp CFrameWnd

```
class CMainWindow : public CFrameWnd {
    public:
          CMainWindow ();
    protected:
          afx_msg void OnPaint ();
          DECLARE_MESSAGE_MAP ()
};
```


Ứng dụng đơn giản – Các thành phần của c.trình…(tt)

- CMainWindow(): hàm khởi tạo cửa sổ giao diện của ứng dụng, override lên hàm chuẩn của lớp CFrameWnd

```
CMainWindow::CMainWindow () {
  Create (NULL, _T ("The Hello Application"));
}
```

- OnPaint: hàm thành phần của lớp CMainWindow, được gọi khi cần cập nhật nội dung cửa sổ. Hàm này được định nghĩa chồng lên hàm chuẩn của lớp CFrameWnd.

```
void CMainWindow::OnPaint () {
  CPaintDC   dc(this);
  CRect              rect;
  GetClientRect (&rect);
  dc.DrawText (_T ("Hello, MFC"), -1, &rect,
              DT_SINGLELINE ¦ DT_CENTER ¦
              DT_VCENTER);
}
```

Ứng dụng đơn giản – Các thành phần của c.trình…(tt)

- Message Map:
  - Làm sao để xử lý 1 message ?
  - MFC dùng Message Map để liên kết các message với những hàm thành phần của lớp cửa sổ
  - Mỗi message sẽ được xử lý bởi 1 hàm thành phần tương ứng

```
BEGIN_MESSAGE_MAP (CMainWindow, CFrameWnd)
      ON_WM_PAINT ()
END_MESSAGE_MAP ()
```

  - ON_WM_PAINT là 1 macro được định nghĩa trong Afxmsg_.h, mặc nhiên liên kết message WM_PAINT với hàm OnPaint

Ứng dụng đơn giản – Các thành phần của c.trình…(tt)

- Xử lý thêm message WM_LBUTTONDOWN
  - Bổ sung thêm 1 hàm thành phần vào khai báo của lớp CMainWindow:

```
afx_msg void OnLButtonDown(UINT nFlags,
                           CPoint point);
```

  - Bổ sung thêm 1 macro vào khai báo Message Map:

```
ON_WM_LBUTTONDOWN ()
```

  - Định nghĩa hàm thành phần **OnLButtonDown**:

```
void CMainWindow::OnLButtonDown(UINT nFlags,
                                CPoint point)
{
       MessageBox("Left button clicked !",
                  "Mouse", MB_OK);
}
```

Ứng dụng đơn giản – Các thành phần của c.trình…(tt)

- Xử lý thêm message WM_MOUSELEAVE
  - Bổ sung thêm 1 hàm thành phần vào khai báo của lớp CMainWindow:

```
afx_msg LRESULT OnMouseLeave();
```

  - Bổ sung thêm 1 macro vào khai báo Message Map:

```
ON_MESSAGE (WM_MOUSELEAVE, OnMouseLeave)
```

  - Định nghĩa hàm thành phần **OnLButtonDown**:

```
LRESULT CMainWindow::OnMouseLeave()
{
   MessageBox("Mouse leaved !", "Mouse", MB_OK);
   return 0;
}
```

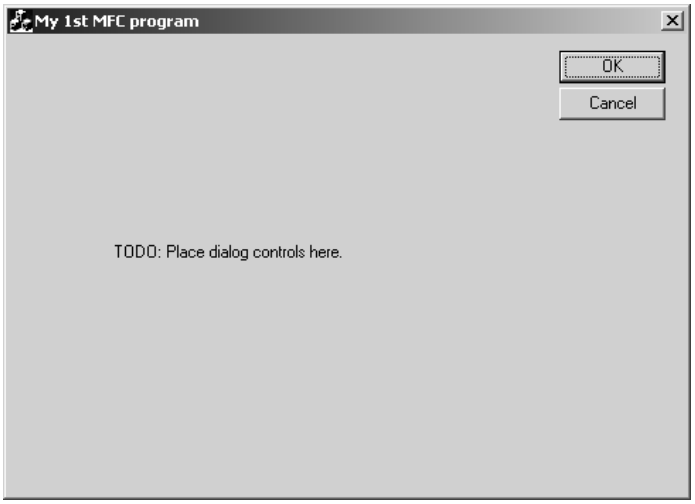

Chương trình MFC đầu tiên – Dialog-based App

- Tạo ứng dụng bằng cách sử dụng MFC AppWizard
- Các thành phần của chương trình

Dialog-based App - Tạo ứng dụng bằng MFC AppWizard

- Chọn menu File → New
- Chọn tab Projects
- Chọn loại project “MFC AppWizard (exe)”
- Đặt tên project và xác định đường dẫn thư mục trong ô “Location”
- Step 1: Chọn loại ứng dụng “Dialog-based”
- Step 2: Chỉ chọn option “3D controls”. Gõ tiêu đề của ứng dụng vào ô “Enter a title…”
- Step 3: chọn theo chế độ mặc định
- Nhấn Finish để kết thúc

Dialog-based App - Tạo ứng dụng bằng MFC AppWizard…(tt)



Ứng dụng MFC (Dialog-based)

Dialog-based App - Các thành phần của chương trình

- Các file chương trình: (xxx là tên project)
  - xxx.h: header file của file xxx.cpp, chứa khai báo lớp CxxxApp để quản lý toàn bộ ứng dụng. Lớp CxxxApp kế thừa từ lớp CWinApp của MFC
  - xxxDlg.h: header file của file xxxDlg.cpp, chứa khai báo lớp CxxxDlg để quản lý cửa sổ Dialog giao diện của ứng dụng. Lớp CxxxDlg kế thừa từ lớp CDialog của MFC
  - Resource.h: header file, chứa các hằng ID của các resource được định nghĩa trong file xxx.rc
  - xxxDlg.cpp: cài đặt các hàm thành phần của lớp CxxxDlg
  - xxx.cpp: cài đặt các hàm thành phần của lớp CxxxApp
  - xxx.rc: mô tả các resource (tài nguyên) của ứng dụng

Dialog-based App- Các thành phần của chương trình…(tt)

- Lớp CxxxDlg:
  - Trong ứng dụng Dialog-based, cửa sổ giao diện chính là 1 Dialog, nên ứng dụng dùng lớp CxxxDlg thay vì lớp CMainWindow

```
class CxxxDlg : public CDialog
{
 public:
        CxxxDlg(CWnd* pParent = NULL);
        enum { IDD = IDD_xxx_DIALOG };
 protected:
        virtual void DoDataExchange(CDataExchange* pDX);
 protected:
        HICON m_hIcon;
 virtual BOOL OnInitDialog();
 afx_msg void OnPaint();
 afx_msg HCURSOR OnQueryDragIcon();
 DECLARE_MESSAGE_MAP()
};
```

Xử lý Mouse và Keyboard

- Xử lý mouse
  - Thông điệp của mouse
  - Ví dụ: Vẽ hình bằng mouse
- Xử lý keyboard
  - Thông điệp của keyboard
  - Ví dụ: Xử lý phím nhấn

Xử lý Mouse

- Thông điệp của mouse
  - WM_LBUTTONDOWN
  - WM_LBUTTONUP
  - WM_LBUTTONDBLCLK
  - WM_RBUTTONDOWN
  - WM_RBUTTONUP
  - WM_RBUTTONDBLCLK
  - WM_MOUSEMOVE
  - WM_MOUSEWHEEL

Xử lý Mouse…(tt)

- Thông điệp của mouse (tt)
  - Với mỗi thông điệp của mouse, Windows gởi kèm 2 tham số wParam và lParam
  - wParam: cho biết phím nào đang được nhấn (Ctrl, Shift)
  - lParam: cho biết toạ độ hiện tại
    - LOWORD(lParam): tọa độ x
    - HIWORD(lParam): tọa độ y

Xử lý Mouse…(tt)

- Ví dụ: Vẽ hình bằng mouse
  - *Mô tả:* khi user nhấn giữ nút trái chuột & di chuyển → vẽ 1 đường thẳng
  - Các xử lý cần thiết:
    - WM_LBUTTONDOWN ⇔ OnLButtonDown
    - WM_MOUSEMOVE ⇔ OnMouseMove
  - Các bước thực hiện:
    - Định nghĩa 2 biến m_PrevX, m_PrevY trong class CxxxDlg
    - Định nghĩa hàm xử lý message WM_LBUTTONDOWN trong class CxxxDlg
    - Định nghĩa hàm xử lý message WM_MOUSEMOVE trong class CxxxDlg

Xử lý Mouse…(tt)

- Vẽ hình bằng mouse…(tt)

```
void CxxxDlg::OnLButtonDown(UINT nFlags,
                            CPoint point)
{
    // TODO: Add your message handler code here
    // and/or call default
    m_PrevX = point.x;
    m_PrevY = point.y;

    CDialog::OnLButtonDown(nFlags, point);
}
```


Xử lý Mouse…(tt)

- Vẽ hình bằng mouse…(tt)

```
void CxxxDlg::OnMouseMove(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here
    if ((nFlags & MK_LBUTTON) == MK_LBUTTON) {
       // Get the Device Context
       CClientDC dc(this);

       // Draw a line from the prev point to current point
       dc.MoveTo(m_StartX, m_StartY);
       dc.LineTo(point.x, point.y);

       // Save the current point as the previous point
       m_PrevX = point.x;
       m_PrevY = point.y;
    }

    CDialog::OnMouseMove(nFlags, point);
}
```

Xử lý keyboard

- Thông điệp của keyboard
  - WM_KEYDOWN / WM_KEYUP: phát sinh khi 1 phím (không phải là phím hệ thống) được nhấn xuống/thả ra
    - Hàm xử lý tương ứng: CWnd::OnKeyDown, CWnd::OnKeyUp
    - wParam: virtual-key code
    - lParam: chứa các thông tin khác (số lần lặp lại phím, scan code, extended key,…)
  - WM_CHAR: là kết quả phát sinh do message WM_KEYDOWN, báo hiệu 1 ký tự in được (printed character) đã được tạo ra
    - Hàm xử lý tương ứng: CWnd::OnChar
    - wParam: mã ký tự
    - lParam: chứa các thông tin khác (số lần lặp lại do nhấn giữ phím, có phím Alt nhấn kèm,…)

Xử lý keyboard…(tt)

- Ví dụ: Xử lý phím nhấn
  - *Mô tả:* khi user nhấn một phím → hiển thị 1 MessageBox thông báo
  - Các xử lý cần thiết
    - WM_KEYDOWN ⇔ OnKeyDown
  - Các bước thực hiện
    - Định nghĩa hàm xử lý message WM_KEYDOWN trong class CxxxDlg

Xử lý menu

- Một vài khái niệm
- Tạo lập menu
- Load và hiển thị menu
- Xử lý khi menu item được chọn
- Thay đổi trạng thái menu
- Ví dụ

Xử lý menu - Một vài khái niệm

- Menu bar: thanh menu. Bao gồm nhiều drop-down menu và menu item
- Drop-down menu: một phần của menu bar, chứa các menu item hoặc các drop-down menu khác. VD. File, Edit, …
- Menu item: tương ứng với 1 lệnh của chương trình. Mỗi menu item được xác định bằng 1 số nguyên phân biệt, gọi là item ID hay command ID. VD. Open, Save, …
- Popup menu: giống như drop-down menu, nhưng có thể xuất hiện ở vị trí bất kỳ trên màn hình (thường khi nhấn nút phải mouse)
- System menu: chứa các lệnh hệ thống điều khiển cửa sổ. VD. Minimize, Maximize, Close, …

Xử lý menu - Tạo lập menu

- Thường có 2 cách chính để tạo menu:
  - Tạo menu ở dạng resource của ứng dụng, và load vào khi chạy
  - Tạo trực tiếp bằng các hàm khi ứng dụng đang chạy.
    - Lớp sử dụng để quản lý menu: CMenu
    - Các hàm thành phần: CreateMenu, InsertMenu, …

Xử lý menu - Tạo lập menu…(tt)

xxx.rc

```
IDR_MAINFRAME MENU PRELOAD DISCARDABLE
BEGIN
        POPUP "&File"
        BEGIN
                MENUITEM "&New\tCtrl+N", ID_FILE_NEW
                MENUITEM "&Open...\tCtrl+O", ID_FILE_OPEN
                MENUITEM SEPARATOR
                MENUITEM "E&xit", ID_APP_EXIT
        END
        POPUP "&Edit"
        BEGIN
                MENUITEM "&Undo\tCtrl+Z", ID_EDIT_UNDO
                MENUITEM SEPARATOR
                MENUITEM "Cu&t\tCtrl+X", ID_EDIT_CUT
                MENUITEM "&Copy\tCtrl+C", ID_EDIT_COPY
                MENUITEM "&Paste\tCtrl+V", ID_EDIT_PASTE
        END
END
```

Xử lý menu - Load và hiển thị menu

- Xác định menu bar khi tạo cửa sổ:

```
Create(NULL, _T("My Application"),
    WS_OVERLAPPEDWINDOW, rectDefault, NULL,
    MAKEINTRESOURCE(IDR_MAINFRAME));
```

- Thay đổi menu bar:

```
CMenu menu;
menu.LoadMenu(IDR_MAINFRAME);
SetMenu(&menu);
menu.Detach();
```


Xử lý menu - Load và hiển thị menu…(tt)

- MAKEINTRESOURCE: macro dùng để chuyển đổi 1 số nguyên (resource ID) thành dạng LPSTR
- CMenu::LoadMenu: load 1 resource menu bar và gán vào đối tượng CMenu
- CWnd::SetMenu: gán menu bar cho 1 cửa sổ
- CMenu::Detach: gỡ bỏ menu bar ra khỏi đối tượng CMenu, để menu bar không bị huỷ bỏ cùng với đối tượng CMenu khi ra khỏi phạm vi khai báo

Xử lý menu - Xử lý khi menu item được chọn

- Các thông điệp của menu
- Xử lý lệnh của menu item
- Nhóm lệnh (Command range)

Xử lý menu - Xử lý khi menu item được chọn…(tt)

- Các thông điệp của menu:
  - WM_MENUSELECT: phát sinh khi user tác động lên menu. Thông điệp này có thể dùng để cập nhật trạng thái của menu (trường hợp menu thay đổi theo ngữ cảnh – Context-sensitive Menu)
    - Hàm xử lý tương ứng: CWnd::OnMenuSelect
    - wParam:
      - LOWORD(wParam): ID của menu item hoặc index của menu popup
      - HIWORD(wParam): các thông tin khác (trạng thái menu, loại menu, …)
    - lParam: handle của menu

Xử lý menu - Xử lý khi menu item được chọn…(tt)

- Các thông điệp của menu: (tt)
  - WM_COMMAND: phát sinh khi user chọn 1 menu item
    - Hàm xử lý tương ứng: CWnd::OnCommand
    - wParam:
      - LOWORD(wParam): ID của menu item hoặc của control
      - HIWORD(wParam): nguồn gốc phát sinh, 1 nếu sinh ra do 1 phím tắt; 0 nếu chọn trực tiếp từ menu
    - lParam:
      - NULL nếu message này phát sinh từ menu
      - Nếu message phát sinh từ 1 control, lParam sẽ chứa handle của control đó

Xử lý menu - Xử lý khi menu item được chọn…(tt)

- Xử lý lệnh của menu item
  - Dựa trên message WM_COMMAND
  - Định nghĩa message map

```
ON_COMMAND (ID_FILE_OPEN, OnMyFileOpen)
ON_COMMAND (ID_FILE_EXIT, OnMyFileExit)
```

  - Viết hàm thành phần xử lý cho menu item tương ứng

```
void CMainFrame::OnMyFileOpen () {
 // Thực hiện thao tác mở file
 …
}
void CMainFrame::OnMyFileExit () {
 PostMessage (WM_CLOSE, 0, 0);
}
```

Xử lý menu - Xử lý khi menu item được chọn…(tt)

- Nhóm lệnh (Command range)
  - Là 1 nhóm menu item hoạt động theo nguyên tắc “Chỉ có 1 phần tử được chọn tại 1 thời điểm”
  - VD. Chức năng vẽ hình “Line / Circle / Rectangle”
  - Cách thức xử lý ?
    - Cách 1: map tất cả xử lý của các menu item này vào chung 1 hàm xử lý
    - Cách 2: dùng macro ON_COMMAND_RANGE

Xử lý menu - Xử lý khi menu item được chọn…(tt)

- Nhóm lệnh (Command range) (tt)
  - Cách 1: map tất cả xử lý của các menu item này vào chung 1 hàm xử lý

```
// Định nghĩa Message map
ON_COMMAND (ID_DRAW_LINE, OnDraw)
ON_COMMAND (ID_DRAW_CIRCLE, OnDraw)
ON_COMMAND (ID_DRAW_RECTANGLE, OnDraw)

// Hàm xử lý chung, xác định item hiện hành
void CMainFrame::OnDraw () {
   m_nCurrentDraw =
      (UINT) LOWORD(GetCurrentMessage()->wParam);
}
```

Xử lý menu - Xử lý khi menu item được chọn…(tt)

- Nhóm lệnh (Command range) (tt)
  - Cách 2: dùng macro ON_COMMAND_RANGE

```
// Định nghĩa Message map
ON_COMMAND_RANGE (ID_DRAW_LINE,
                  ID_DRAW_RECTANGLE, OnDraw)

// Hàm xử lý chung, xác định item hiện hành
void CMainFrame::OnDraw (UINT nID) {
      m_nCurrentDraw = nID;
}
```


Xử lý menu - Thay đổi trạng thái menu

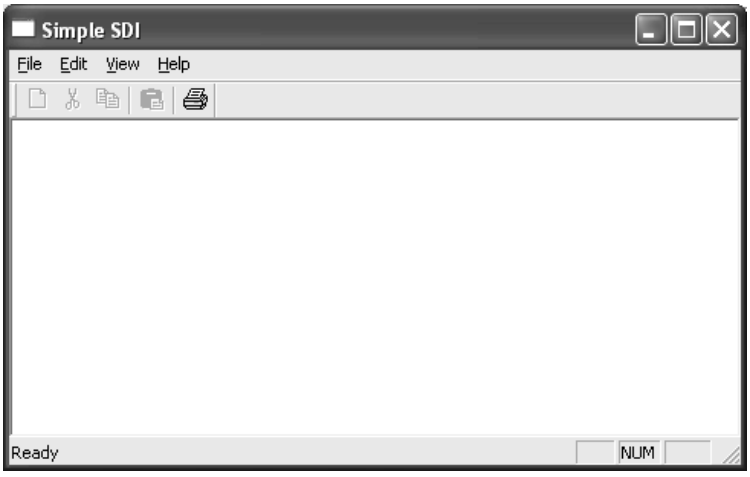
- Các ví dụ:
  - Khi user chọn chức năng vẽ Circle → cần thể hiện 1 dấu check (☑) phía trước
  - Chức năng Cut/Copy/Delete chỉ được kích hoạt khi user đánh dấu chọn 1 đoạn text
  - Chức năng Paste chỉ được kích hoạt khi clipboard khác rỗng
  - ...
- Cách xử lý

```
void CMainFrame::OnDraw(UINT nID) {
    CMenu* pMenu = GetMenu();
    pMenu->CheckMenuItem(m_nCurrentDraw,MF_UNCHECKED);
    m_nCurrentDraw = nID;
    pMenu->CheckMenuItem(m_nCurrentDraw, MF_CHECKED);
}
```

Xử lý menu – Ví dụ

- Tạo 1 ứng dụng SDI
  - Chọn menu File → New
  - Chọn tab Projects
  - Chọn loại project “MFC AppWizard (exe)”
  - Đặt tên project và xác định đường dẫn thư mục trong ô “Location”
  - Step 1: Chọn loại ứng dụng “Single Document”, bỏ option “Document/View architecture support”
  - Nhấn Finish để kết thúc

Xử lý menu – Ví dụ…(tt)



Xử lý menu – Ví dụ…(tt)

- Xử lý lệnh của menu item
  - Vẽ thêm vào menu popup File các item: New, Open, Save
  - Định nghĩa Message Map cho các hàm xử lý item

```
ON_COMMAND(ID_FILE_NEW, OnFileNew)
ON_COMMAND(ID_FILE_OPEN, OnFileOpen)
ON_COMMAND(ID_FILE_SAVE, OnFileSave)
```


Xử lý menu – Ví dụ…(tt)

- Xử lý lệnh của menu item (tt)
  - Viết xử lý lệnh cho từng item

```
void CMainFrame::OnFileNew()
{
   // TODO: Add your command handler code here
   MessageBox("Ban vua chon item New", "File");
}
void CMainFrame::OnFileOpen()
{
   // TODO: Add your command handler code here
   MessageBox("Ban vua chon item Open", "File");
}
void CMainFrame::OnFileSave()
{
   // TODO: Add your command handler code here
   MessageBox("Ban vua chon item Save", "File");
}
```

Xử lý menu – Ví dụ…(tt)

- Xử lý chọn nhóm lệnh
  - Vẽ thêm menu popup Draw với các item: Line, Circle, Rectangle
  - Định nghĩa message map

```
ON_COMMAND_RANGE (ID_DRAW_LINE,
                  ID_DRAW_RECTANGLE, OnDraw)
```

  - Viết hàm xử lý

```
void CMainFrame::OnDraw(UINT nID) {
        CMenu* pMenu = GetMenu();
        pMenu->CheckMenuItem(m_nCurrentDraw,
                    MF_UNCHECKED);
        m_nCurrentDraw = nID;
        pMenu->CheckMenuItem(m_nCurrentDraw,
                    MF_CHECKED);
}
```

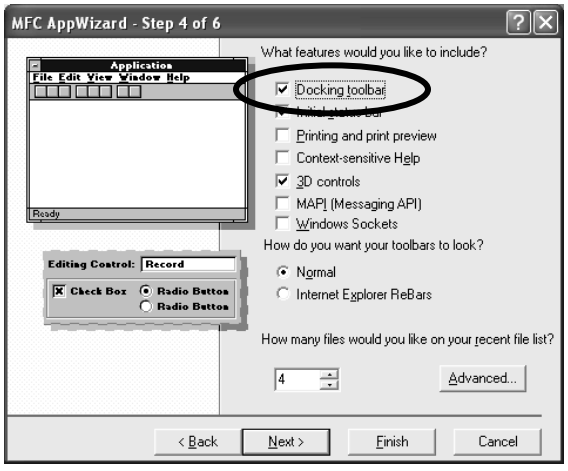

Toolbar

- Tạo một ứng dụng có Toolbar bằng AppWizard
- Tạo Toolbar bằng lớp CToolBar

Toolbar - Tạo một ứng dụng bằng AppWizard

- Tạo một ứng dụng có Toolbar bằng AppWizard
  - Chọn menu File → New
  - Chọn tab Projects
  - Chọn loại project “MFC AppWizard (exe)”
  - Đặt tên project và xác định đường dẫn thư mục trong ô “Location”
  - Step 1: Chọn loại ứng dụng “Single Document”, bỏ option “Document/View architecture support”
  - Nhấn Finish để kết thúc

Toolbar - Tạo một ứng dụng bằng AppWizard…(tt)



Chọn option này để AppWizard tự động tạo ra một Docking Toolbar

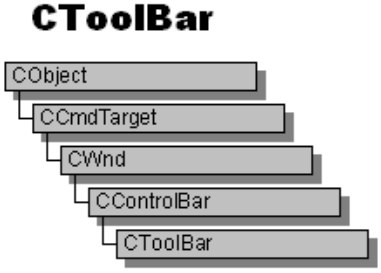
Toolbar - Tạo một ứng dụng bằng AppWizard…(tt)

- Các xử lý trong hàm OnCreate của lớp CMainFrame

```
if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT,
          WS_CHILD | WS_VISIBLE | CBRS_TOP|
          CBRS_GRIPPER | CBRS_TOOLTIPS | CBRS_FLYBY |
          CBRS_SIZE_DYNAMIC) ||
    !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
{
    TRACE0("Failed to create toolbar\n");
    return -1;      // fail to create
}
// Xác định thuộc tính Docking
m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
EnableDocking(CBRS_ALIGN_ANY);
DockControlBar(&m_wndToolBar);
```


Toolbar - Tạo Toolbar bằng lớp CToolBar

- Tạo lập và hiển thị
- Ẩn/hiện thanh ToolBar
- Thêm các ToolTip và FlyBy text



Toolbar - Tạo Toolbar bằng lớp CToolBar…(tt)

- Tạo lập và hiển thị:
  - Bước 1: thiết kế DrawToolBar bằng RC editor, bao gồm các chức năng: Line, Circle, Rectangle, có ID là IDR_DRAWTOOLBAR

  - Bước 2: trong class CMainFrame, định nghĩa biến quản lý DrawToolBar

```
// class CMainFrame
CToolBar  m_wndDrawToolBar;
```


Toolbar - Tạo Toolbar bằng lớp CToolBar…(tt)

- Bước 3: trong hàm OnCreate của lớp CMainFrame, viết lệnh tạo lập DrawToolBar

```
// Trong hàm CMainFrame::OnCreate
if (!m_wndDrawToolBar.Create(this) ||
    !m_wndDrawToolBar.LoadToolBar(IDR_DRAWTOOLBAR))
{
    TRACE0("Khong the tao duoc DrawToolBat\n");
    return -1;
}
// Xác định tính chất của ToolBar
m_wndDrawToolBar.SetBarStyle(
    m_wndDrawToolBar.GetBarStyle() | CBRS_TOOLTIPS |
    CBRS_FLYBY | CBRS_SIZE_DYNAMIC);
// Xác định tính chất Docking
m_wndDrawToolBar.EnableDocking(CBRS_ALIGN_ANY);
// Docking toolbar
DockControlBar(&m_wndDrawToolBar);
```

Toolbar - Tạo Toolbar bằng lớp CToolBar…(tt)

- Ẩn/hiện thanh ToolBar
  - Cách thực hiện:
    - Thêm 1 menu item mới vào menu popup View, với ID là ID_VIEW_DRAWTOOLBAR
    - Viết hàm xử lý cho menu item này

```
void CMainFrame::OnViewDrawtoolbar()
{
  // TODO: Add your command handler code here
  BOOL  bVisible = m_wndDrawToolBar.GetStyle()
                        & WS_VISIBLE;
  ShowControlBar(&m_wndDrawToolBar, !bVisible,
                        FALSE);

  CMenu* pMenu = GetMenu();
  pMenu->CheckMenuItem(ID_VIEW_DRAWTOOLBAR,
    (!bVisible==1) ? MF_CHECKED : MF_UNCHECKED);
}
```


Toolbar - Tạo Toolbar bằng lớp CToolBar…(tt)

- Thêm các ToolTip và FlyBy text
  - ToolTip là 1 cửa sổ nhỏ chứa câu giải thích ngắn về công dụng của 1 button trên ToolBar

  - FlyBy text là 1 câu thông báo được hiển thị trên StatusBar khi user di chuyển mouse đến 1 button của ToolBar

Toolbar - Tạo Toolbar bằng lớp CToolBar…(tt)

- Thêm các ToolTip và FlyBy text (tt)
  - Cách thực hiện:
    - ToolBar phải có thuộc tính CBRS_TOOLTIPS ¦ CBRS_FLYBY
    - Tạo 1 bảng mô tả chuỗi (StringTable)
    - ID của chuỗi trùng với ID của các button trên ToolBar
    - Chuỗi có thể gồm 2 phần:

```
<FlyBy Text>\n<ToolTip>
VD.
STRINGTABLE DISCARDABLE
BEGIN
   ID_DRAW_LINE        "Draw a line\nLine"
   ID_DRAW_CIRCLE      "Draw a circle\nCircle"
   ID_DRAW_RECTANGLE   "Draw a rect\nRectangle"
END
```

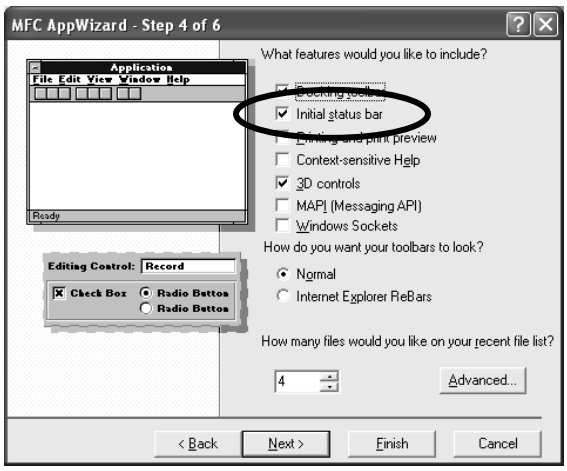

Statusbar

- Tạo một ứng dụng có Statusbar bằng AppWizard
- Tạo Statusbar bằng lớp CStatusBar

Statusbar - Tạo một ứng dụng bằng AppWizard

- Tạo một ứng dụng có Statusbar bằng AppWizard
  - Chọn menu File → New
  - Chọn tab Projects
  - Chọn loại project “MFC AppWizard (exe)”
  - Đặt tên project và xác định đường dẫn thư mục trong ô “Location”
  - Step 1: Chọn loại ứng dụng “Single Document”, bỏ option “Document/View architecture support”
  - Nhấn Finish để kết thúc

Statusbar - Tạo một ứng dụng bằng AppWizard…(tt)



Chọn option này để AppWizard tự động tạo ra một Statusbar

Statusbar - Tạo một ứng dụng bằng AppWizard…(tt)

- Các xử lý tương ứng

```
// Định nghĩa các vùng trên Statusbar
// (file MainFrm.cpp)
static UINT indicators[] =
{
    ID_SEPARATOR,           // status line indicator
    ID_INDICATOR_CAPS,
    ID_INDICATOR_NUM,
    ID_INDICATOR_SCRL,
};
// Tạo lập Statusbar (hàm OnCreate của lớp CMainFrame)
if (!m_wndStatusBar.Create(this) ||
    !m_wndStatusBar.SetIndicators(indicators,
            sizeof(indicators)/sizeof(UINT)))
{
    TRACE0("Không thể tạo được Statusbar\n");
    return -1;      // fail to create
}
```


Statusbar - Tạo Statusbar bằng lớp CStatusBar

- Tạo lập và hiển thị
- Ẩn/hiện Statusbar
- Thể hiện giúp đỡ cho các menu item
- Phân vùng trên Statusbar