

Bài thực hành 2

LẬP TRÌNH HỆ THỐNG CHAT ĐƠN GIẢN BẰNG WINSOCK TRONG MÔI TRƯỜNG LẬP TRÌNH VISUAL C++

Nội dung

4.1 Giới thiệu môi trường lập trình Visual C++ 6.0

4.2 Lập trình Winsock trong VC++

4.3 Thiết kế ứng dụng mạng MiniChat

4.4 Hiện thực chương trình MiniChatServer

4.5 Hiện thực chương trình MiniChatClient

Giới thiệu môi trường lập trình Visual C++ 6.0 (VC++)

- Là môi trường lập trình C++ cho phép thiết kế trực quan giao diện.
 - Các ứng dụng được tổ chức theo dạng project, một project chứa các file khác nhau về mã chương trình, giao diện, các file header...
 - Có nhiều loại ứng dụng trong VC++. Chương này giới thiệu về ứng dụng MFC
-

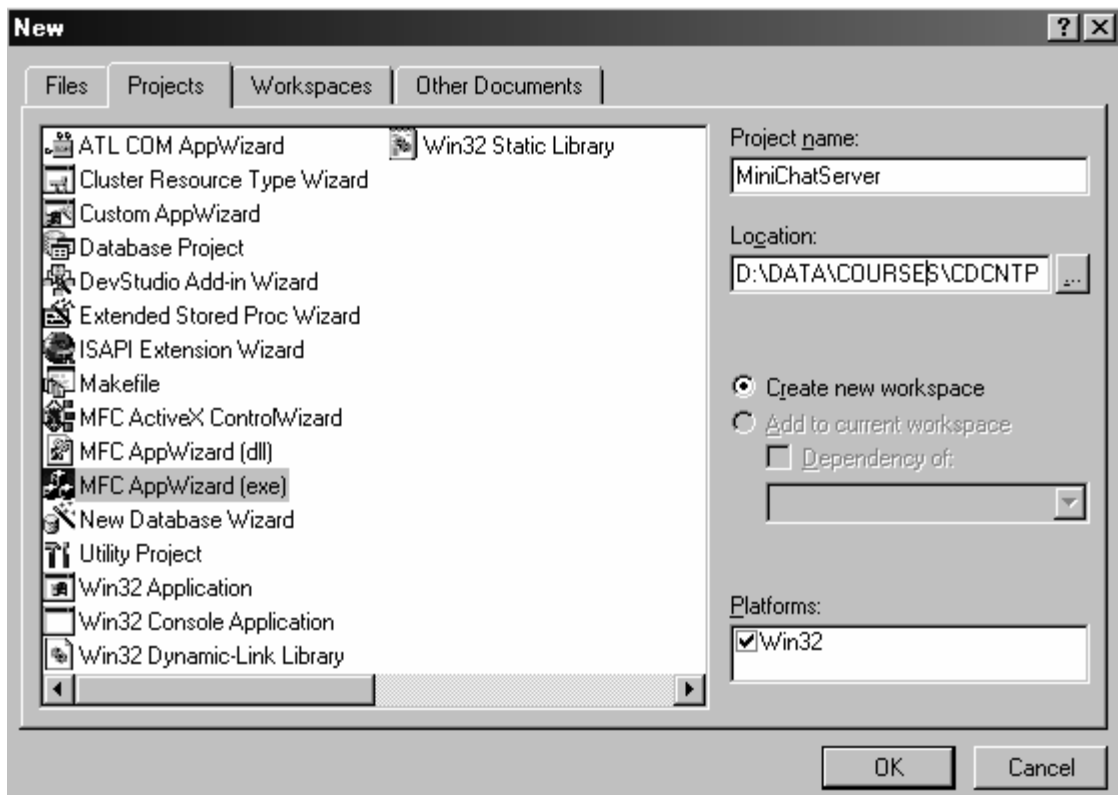
Tạo mới một project

Dùng menu File → New. Hộp thoại như bên dưới xuất hiện

Ở tag projectst, chọn loại ứng dụng là MFC AppWizard (exe).

Ở phần location, chọn thư mục để chứa project.

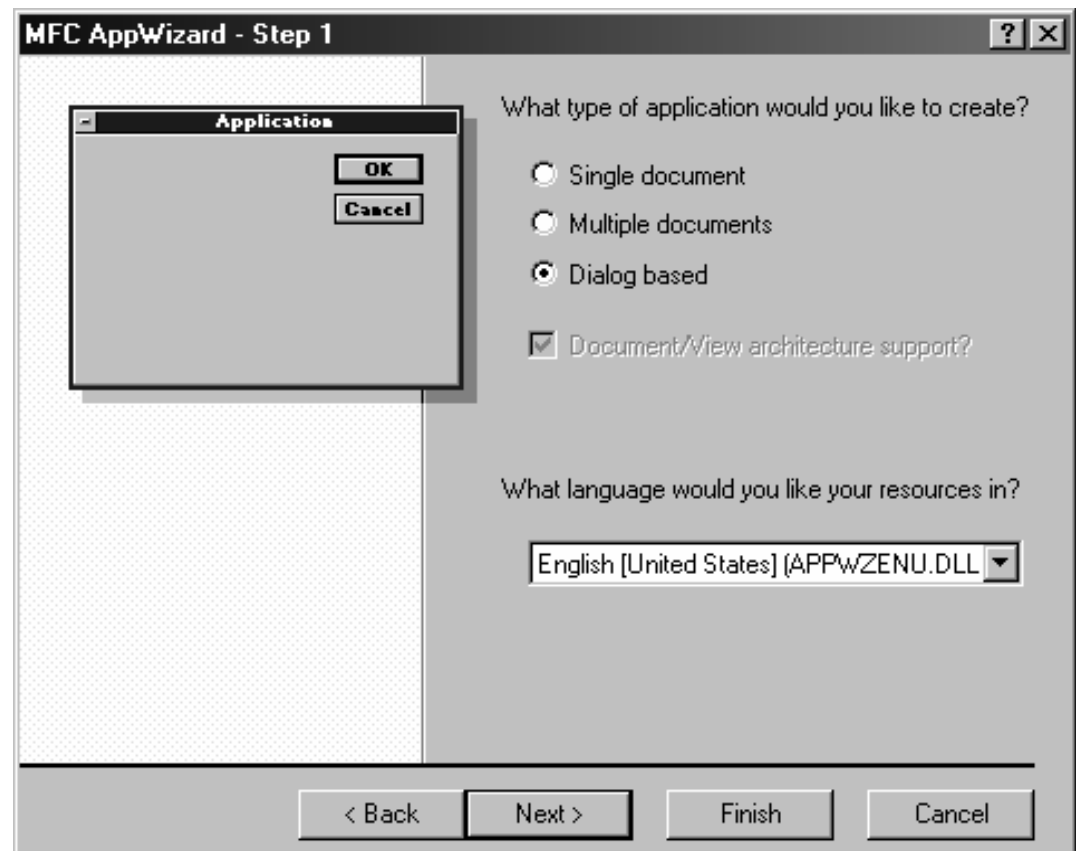
Gõ tên project và chọn OK



Hiệu chỉnh các thông số

Bước thứ nhất chọn loại ứng dụng, chọn dạng Dialog based như hình bên.

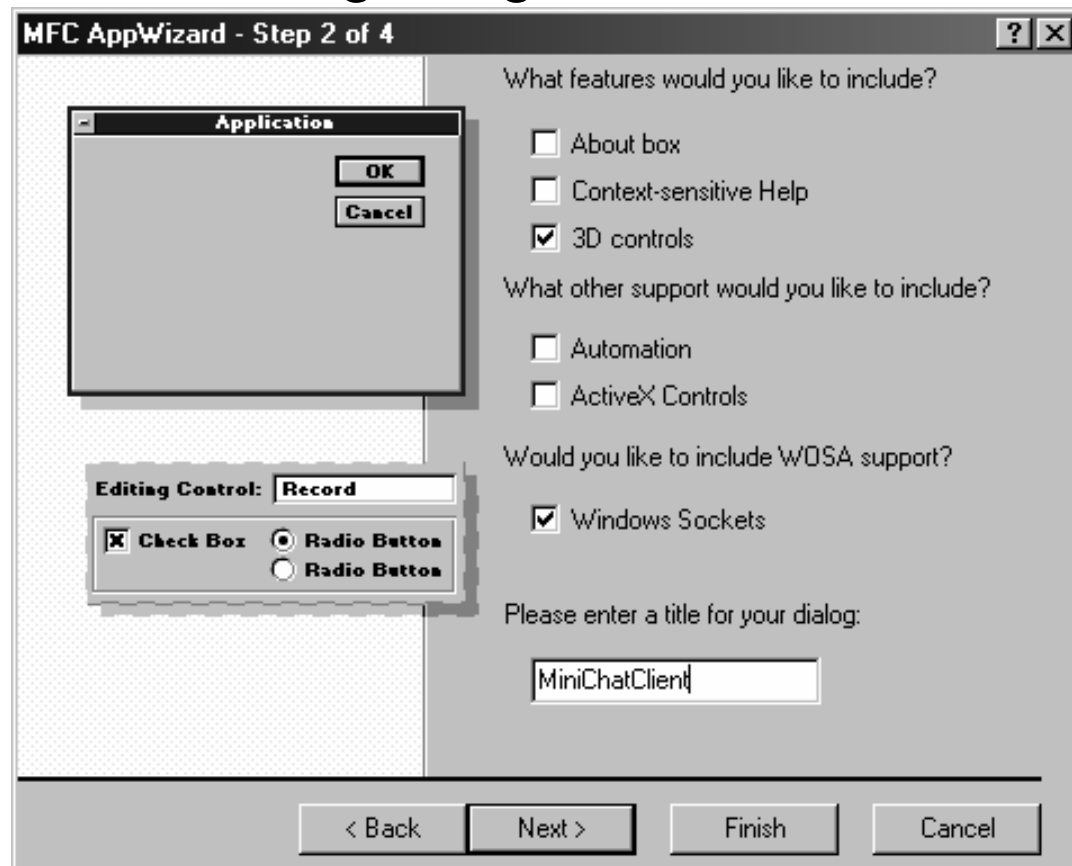
Nhấn button Next để tiếp tục



Hiệu chỉnh các thông số

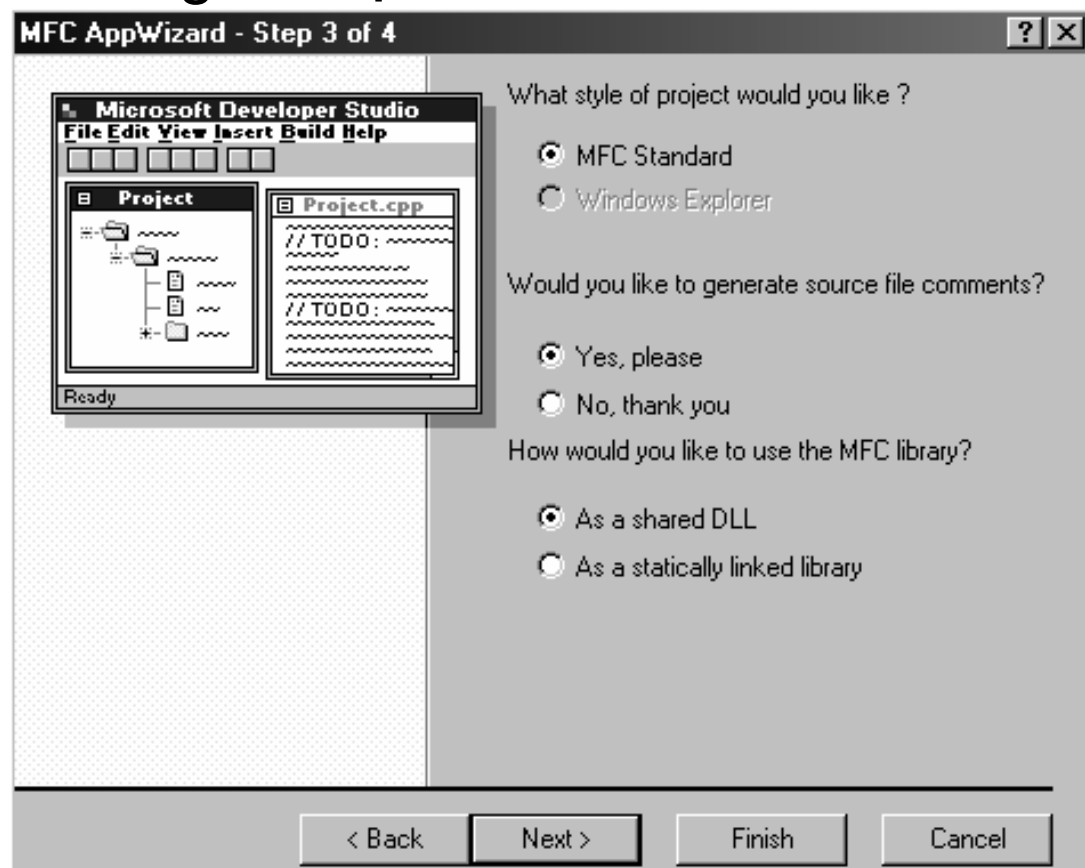
Bước 2, chọn các đặc tính của ứng dụng như hình
+ Phải chọn
checkbox
Windows Sockets

Nhấn button Next
để tiếp tục



Hiệu chỉnh các thông số

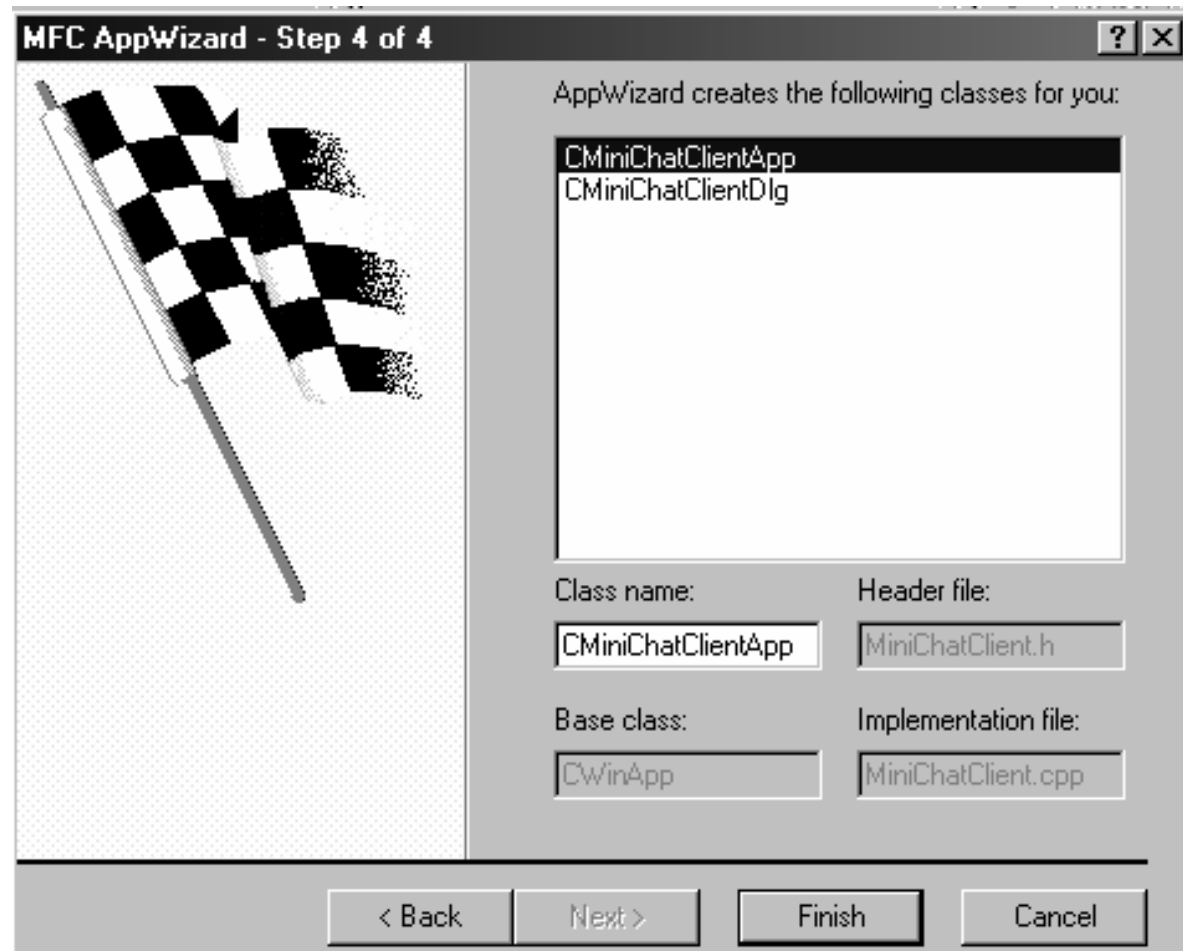
Bước 3, chọn các chức năng hỗ trợ như hình vẽ
Nhấn button Next
để tiếp tục



Hiệu chỉnh các thông số

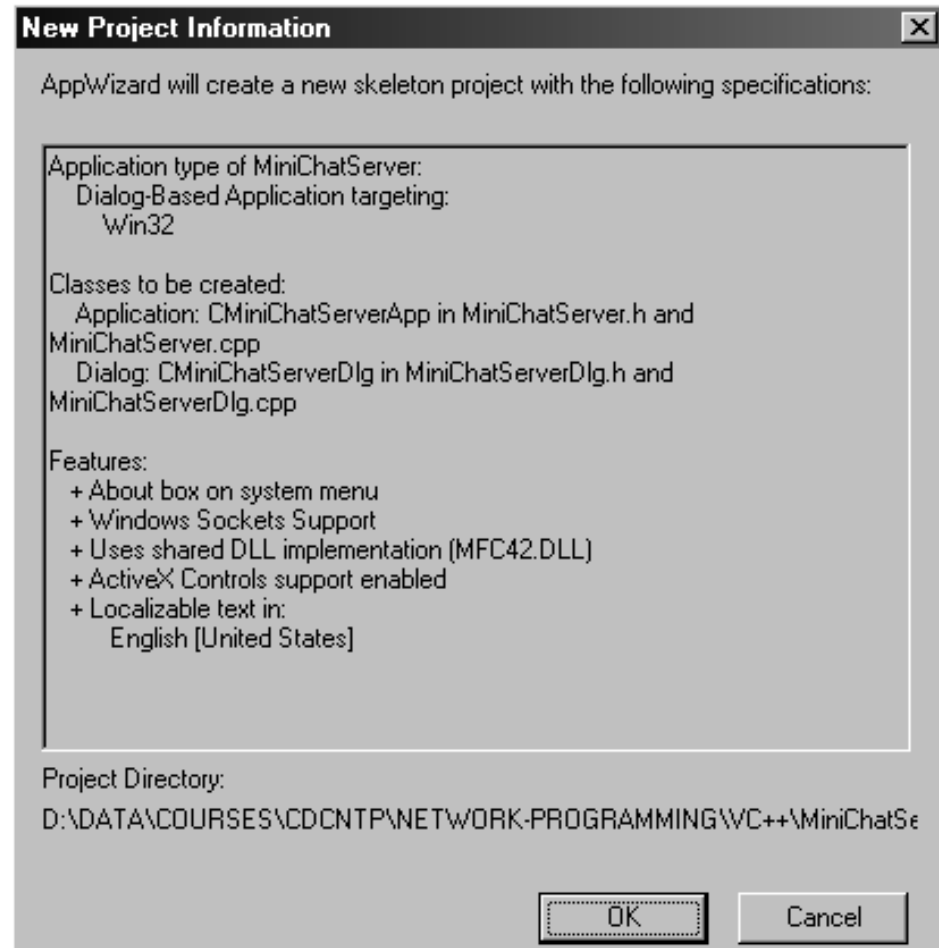
Bước 4: xác nhận các thông số đã chọn. Có thể qua lại các bước trước đó để hiệu chỉnh bằng button Back.

Chọn button Finish để kết thúc

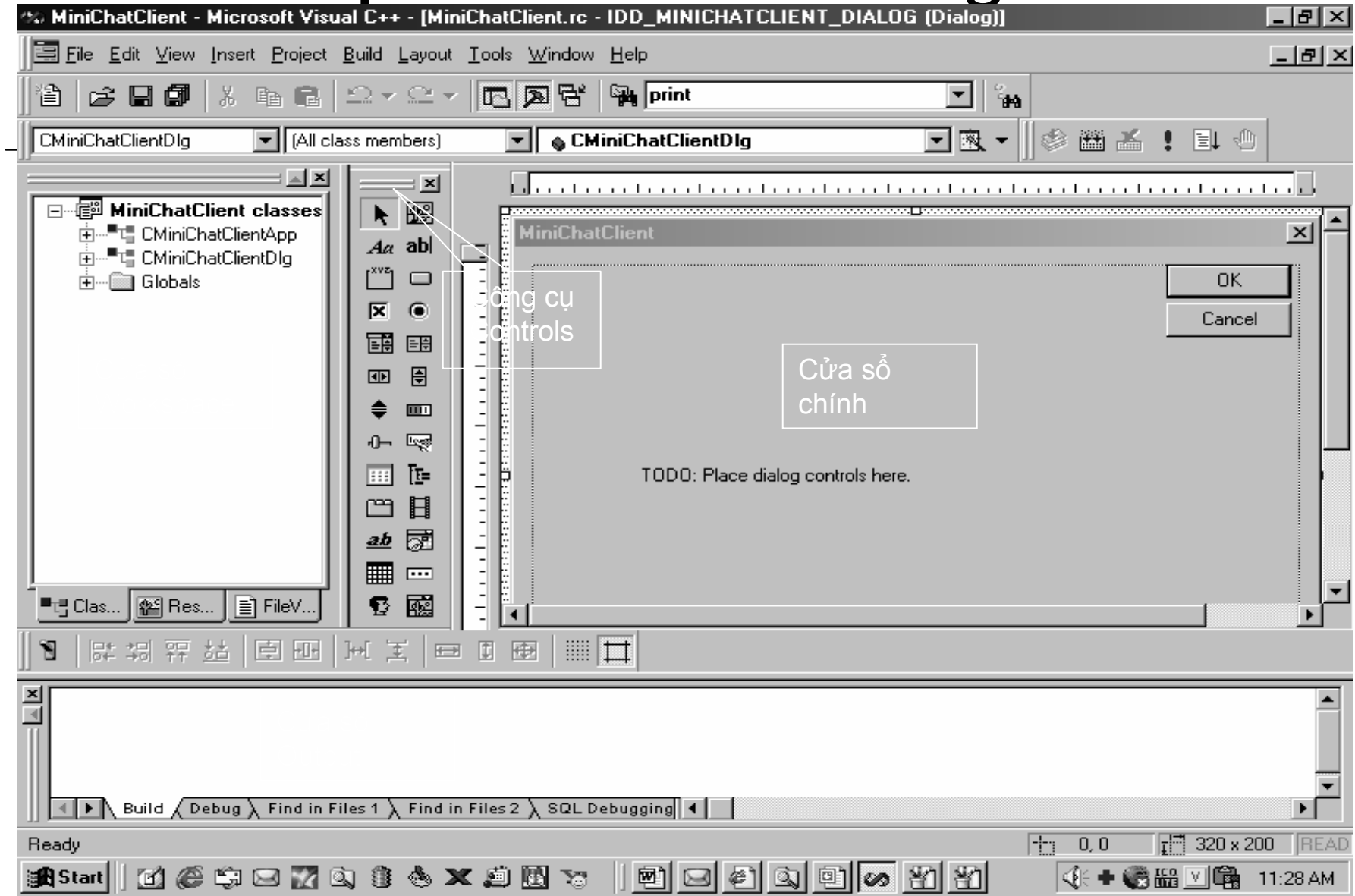


Hiệu chỉnh các thông số





Bước cuối cùng: xác nhận và chọn OK để bắt đầu lập trình

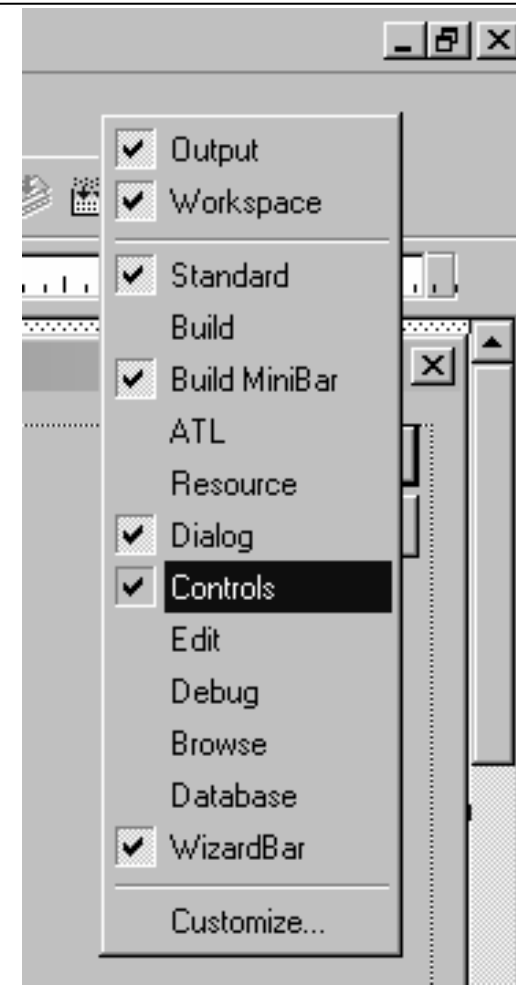


Giao diện của môi trường VC++



Thiết kế giao diện

- Để thiết kế giao diện, ta cần dùng thanh công cụ Controls (right-click vào các thanh công cụ, chọn Controls như hình bên)
- Các đối tượng giao diện thường dùng:
 - Static Text 
 - Edit Box 
 - Button 
 - Listbox 



Vẽ các đối tượng giao diện

- Mở Dialog cần vẽ các đối tượng giao diện (Ở cửa sổ Workspace, chọn chế độ ResourceView, click chọn thư mục dialog, chọn Dialog tương ứng)
 - Muốn vẽ đối tượng giao diện nào click vào đối tượng giao diện đó, đưa trỏ chuột vào Dialog để vẽ (dùng cơ chế Drag chuột, vừa nhấn chuột trái vừa kéo)
-

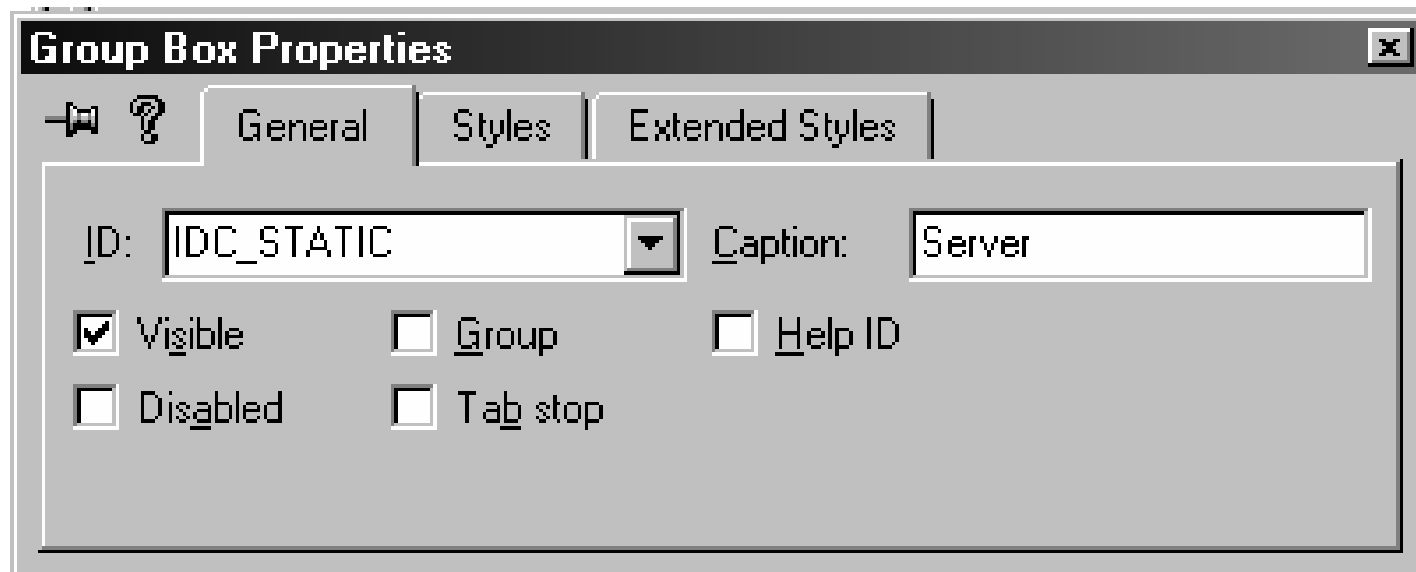
Thiết lập thuộc tính cho các đối tượng giao diện

- Right-click vào đối tượng giao diện và chọn Properties
- ID là thuộc tính tên nhận dạng của đối tượng giao diện
- Tùy mỗi loại đối tượng giao diện có các thuộc tính riêng



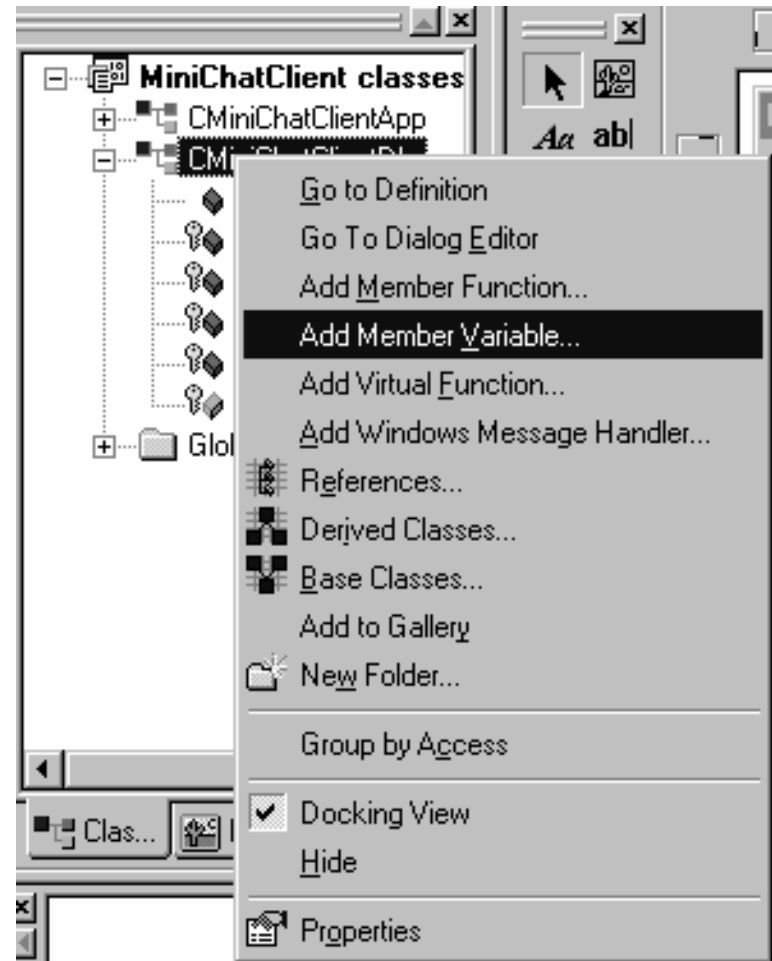
Thiết lập thuộc tính cho các đối tượng giao diện

- Thiết lập caption (Nội dung hiển thị lên phần tử giao diện) cho đối tượng giao diện Button và Static Text như hình bên dưới



Khai báo biến và định nghĩa hàm

- Trong cửa sổ workspace, chọn tab ClassView, right-click vào class C*Dlg, menu hiển thị như hình vẽ bên
- Chọn chức năng Add Member Variable
- Chức năng này cũng dùng tương tự cho việc định nghĩa hàm

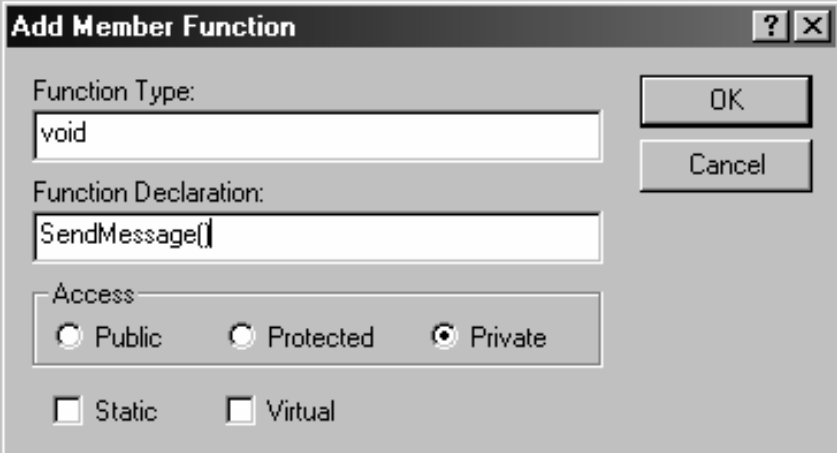


Khai báo biến và định nghĩa hàm

- Khai báo biến như hình trên: đánh kiểu biến, tên biến và tầm vực của biến rồi nhấn OK
- Định nghĩa hàm như hình bên dưới: kiểu trả về, tên hàm và các thông số, tầm vực truy xuất



The 'Add Member Variable' dialog box is shown. It has a title bar with a question mark and a close button. The 'Variable Type' field contains 'SOCKET'. The 'Variable Name' field contains 'ServerSocket'. The 'Access' section has three radio buttons: 'Public' (selected), 'Protected', and 'Private'. There are 'OK' and 'Cancel' buttons on the right.

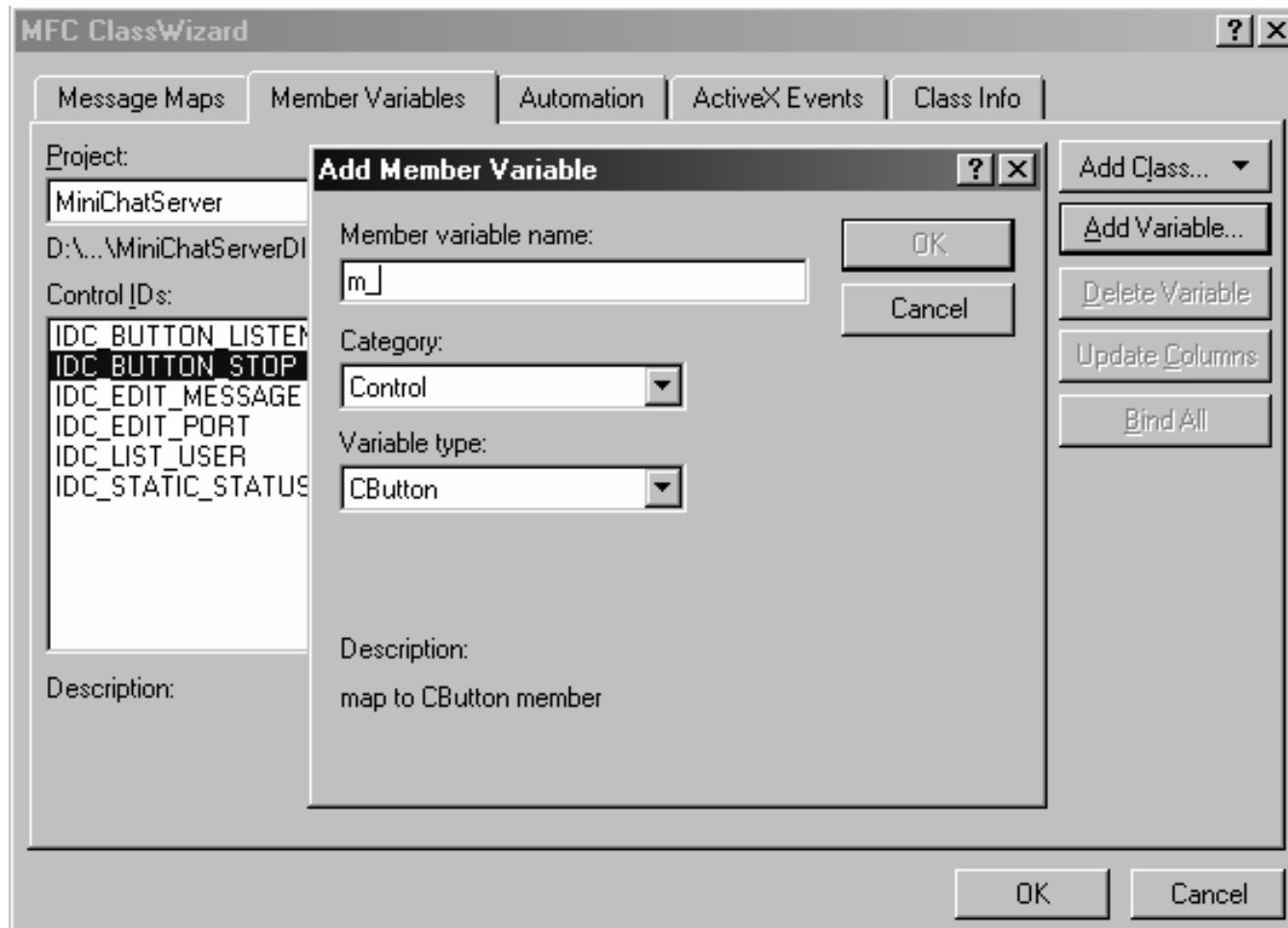


The 'Add Member Function' dialog box is shown. It has a title bar with a question mark and a close button. The 'Function Type' field contains 'void'. The 'Function Declaration' field contains 'SendMessage()'. The 'Access' section has three radio buttons: 'Public', 'Protected', and 'Private' (selected). There are also checkboxes for 'Static' and 'Virtual', both of which are unchecked. There are 'OK' and 'Cancel' buttons on the right.

Gán biến cho đối tượng giao diện

- Mỗi đối tượng giao diện đều có thể truy xuất thông qua biến được định nghĩa
 - Chọn menu View -> ClassWinzard -> Member Variables
 - Chọn đối tượng giao diện tương ứng (nhờ vào ID đã đặt), click button Add Variable)
 - Đặt tên biến, loại biến (Control hoặc Value) và kiểu dữ liệu
-

Gán biến cho đối tượng giao diện



Thiết lập - lấy giá trị phần tử giao diện Edit Box và Static Text

- Thiết lập:
 - Gán giá trị cho biến tương ứng.
 - Dùng lệnh: `UpdateData(FALSE);`
- Lấy giá trị:
 - Dùng lệnh: `UpdateData(TRUE);`
 - Giá trị được truyền cho biến tương ứng của phần tử giao diện

Ví dụ:

```
m_mes=m_mes+"Accepted a connection!\r\n";  
UpdateData ( FALSE ) ;
```

Thêm - loại giá trị cho phần tử giao diện Listbox

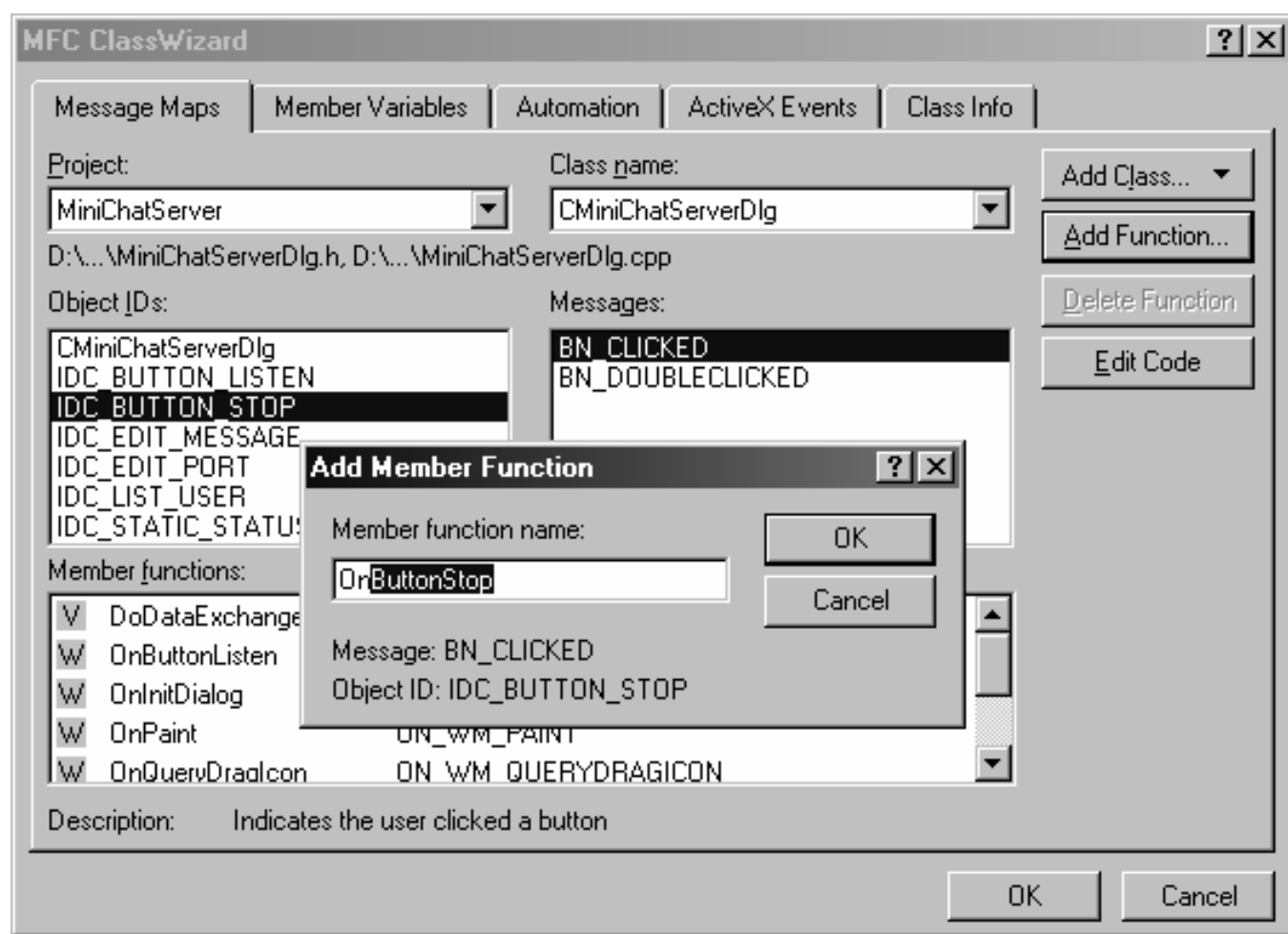
- Thêm vào ListBox:
 - Dùng phương thức `AddString(String)` của đối tượng điều khiển ListBox
 - Loại phần tử ra khỏi ListBox:
 - Dùng phương thức `RemoveString(int index)` của đối tượng điều khiển
 - Lấy index của một phần tử nào, ta cần phải quản lý danh sách của Listbox
-

Tạo hàm xử lý sự kiện cho button

- Khi người sử dụng click chuột vào button nào trên giao diện, hệ thống sẽ sinh ra sự kiện `BN_CLICKED` cho đối tượng đó.
 - Người lập trình phải viết mã để xử lý sự kiện đó.
 - Để tạo hàm xử lý sự kiện, ta có thể double-click vào button, VC++ sẽ đề nghị tên hàm, nhấn OK để bắt đầu viết mã
-

Tạo hàm xử lý sự kiện cho button

Có thể dùng ClassWizard, chọn đối tượng Button, chọn message BN_CLICKED và nhấn Add Function



Lập trình Winsock trong VC++

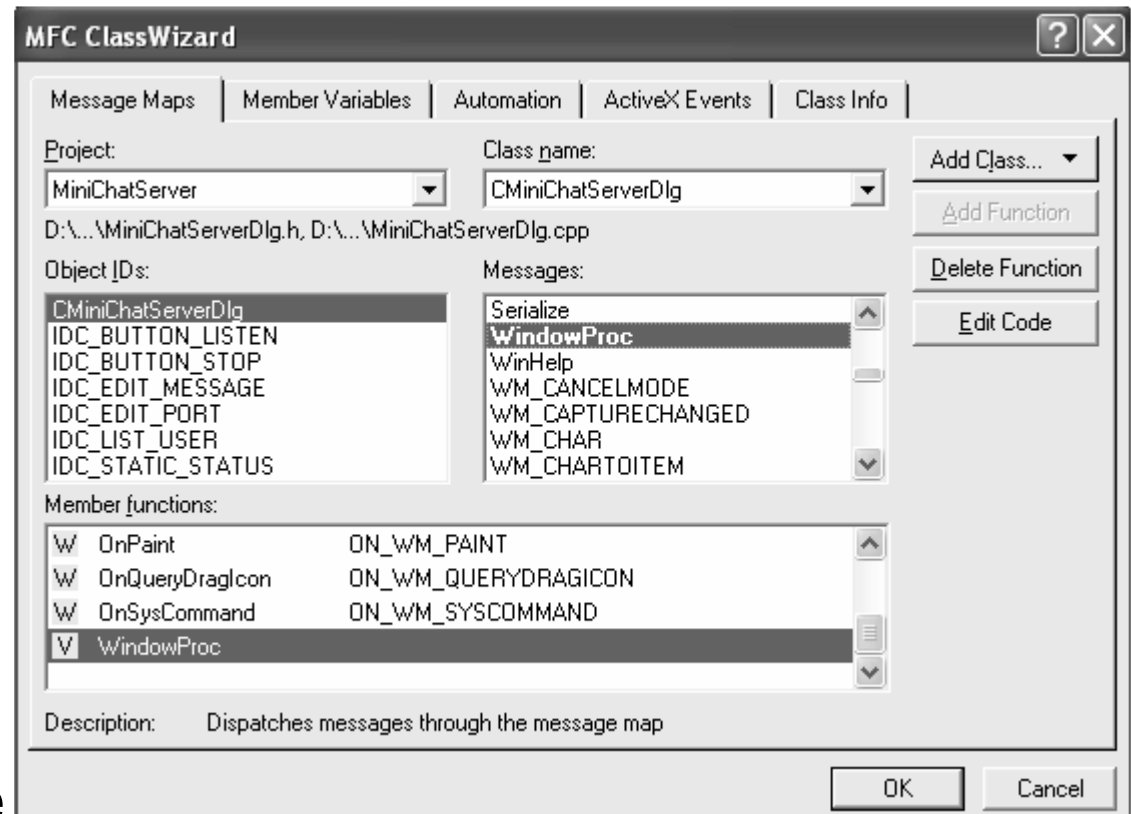
- Phân tích và thiết kế giao diện cần thiết cho ứng dụng mạng.
 - Thiết lập các biến dữ liệu tương ứng với các phần tử giao diện Listbox, Edit box, Static Text
 - Đặt các biến dùng cho lập trình socket
 - Viết mã lệnh trình tự các hàm như đã trình bày ở chương 3
-

Lập trình Winsock trong VC++

- Hàm được gọi đầu tiên khi ứng dụng khởi tạo là OnInitDialog(): chúng ta có thể viết hàm khởi tạo socket, bind, listen, accept trong hàm này
 - Có thể tạo các button để xử lý gọi các hàm nêu trên.
 - Nên tạo các hàm để xử lý sự kiện và các lệnh tương ứng
-

Lập trình Winsock trong VC++

- Tạo hàm **WindowProc** để xử lý các sự kiện mạng:
 - Cửa sổ ClassWinzard, chọn ID là C*Dlg, Messages là WindowProc và click button Add Function
 - Click button Edit code để viết mã



Lập trình Winsock trong VC++

- Một số messages cần quan tâm khi lập trình mạng trong VC++
 - WM_CLOSE: xảy ra khi người sử dụng đóng chương trình
 - WM_KEYUP: xảy ra khi người sử dụng nhả một phím, có thể dùng để detect phím Enter
 - Việc xử lý các messages này cũng cần phải tạo hàm xử lý tương ứng như slide trước
-

Thiết kế ứng dụng mạng MiniChat

- Ứng dụng MiniChat có hai chương trình MiniChatServer và MiniChatClient
 - Trong hệ thống, chỉ có một chương trình server và nhiều chương trình client đang chạy.
 - Chương trình client gửi dữ liệu đến cho chương trình server để yêu cầu thông tin hoặc gửi thông tin => Định nghĩa các loại dữ liệu gửi
-

Định nghĩa các loại dữ liệu gửi

- Dữ liệu gửi của client cho server:
 - Tham gia vào chat room: LOGIN:nickname*
 - nickname là tên của người sử dụng dùng để chat, không được trùng với các nickname khác
 - Gửi message cho toàn bộ chat room: PUBLIC: nicknamesender:message*
 - Gửi message cho riêng một user: PRIVATE: nicknamesender: nicknamereciever: message*
 - Thoát khỏi chat room: QUIT*
-

Định nghĩa các loại dữ liệu gửi

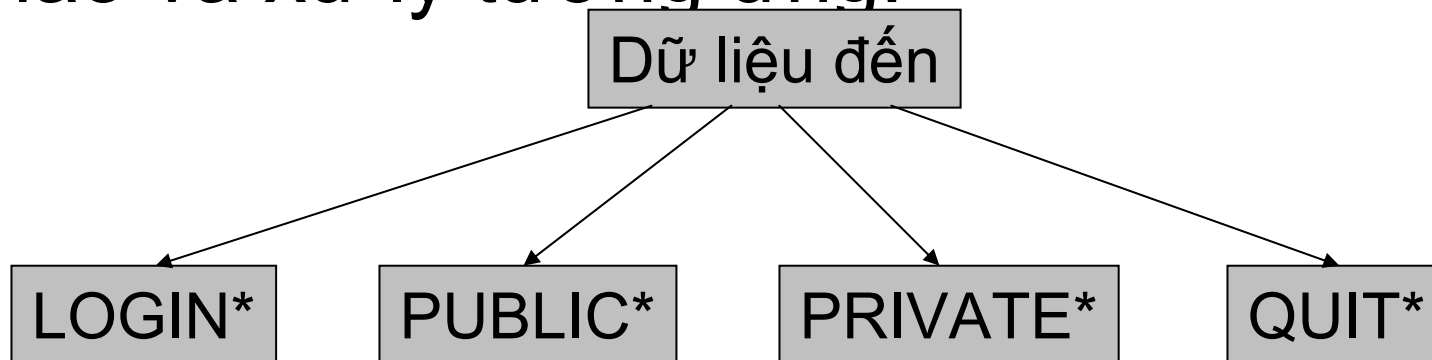
- Dữ liệu gửi từ server cho client:
 - Danh sách các user (nickname) có trong chatroom: LIST[:nickname]+*
 - Ví dụ: LIST:cuc:mai:lan:dao*
 - Message cho toàn bộ user:
PUBLIC:nicknamesender:message*
 - Message cho riêng một user:
PRIVATE:nicknamesender:message*
-

Định nghĩa các loại dữ liệu gửi

- Dữ liệu gửi từ server cho client:
 - Một user login vào: USERL:nickname*
 - Đã xử lý yêu cầu đúng: +OK
 - Các lỗi:
 - -100: Unknown command
 - -101: Not login
 - -102: Nickname existed
 - -103: Nickname not exist
 - -104: Cannot send the message
 - -105: Not accept null nickname
 - -106: Login already
-

Thiết kế sơ đồ chức năng của ứng dụng MiniChatServer

- Chương trình **server** mở socket và lắng nghe kết nối từ các client. Khi có dữ liệu đến, server phân tích dữ liệu thuộc dạng nào và xử lý tương ứng:

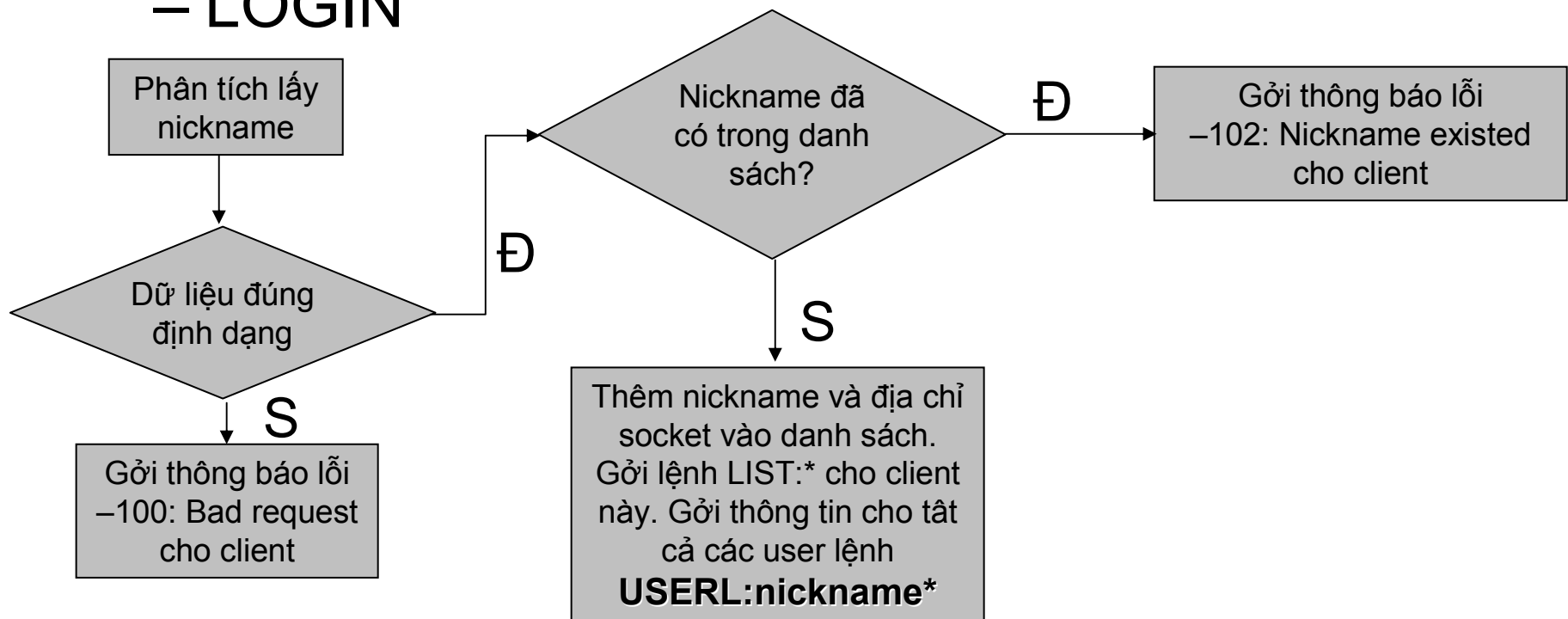


- Nếu không thuộc các định dạng trên thì gửi lệnh **-100: Bad request** cho client
-

Thiết kế sơ đồ chức năng của ứng dụng MiniChatServer

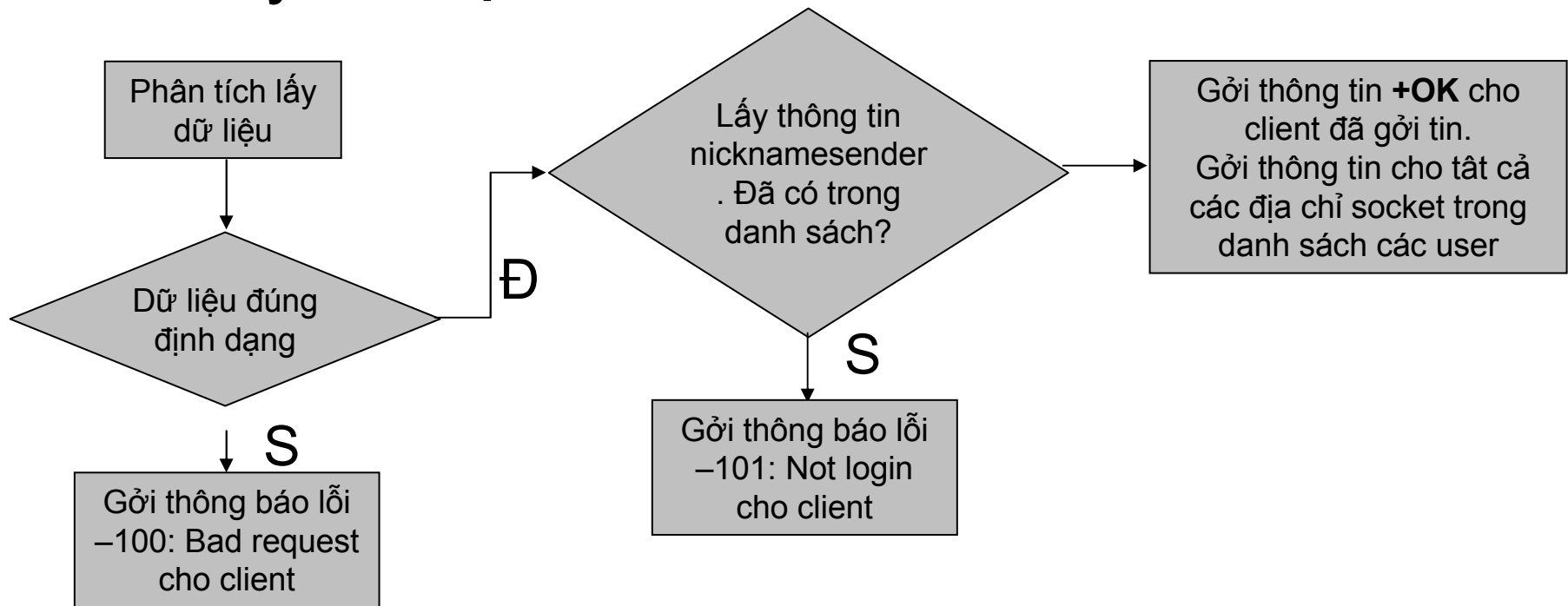
- Với mỗi loại dữ liệu sẽ xử lý tương ứng:

– LOGIN*



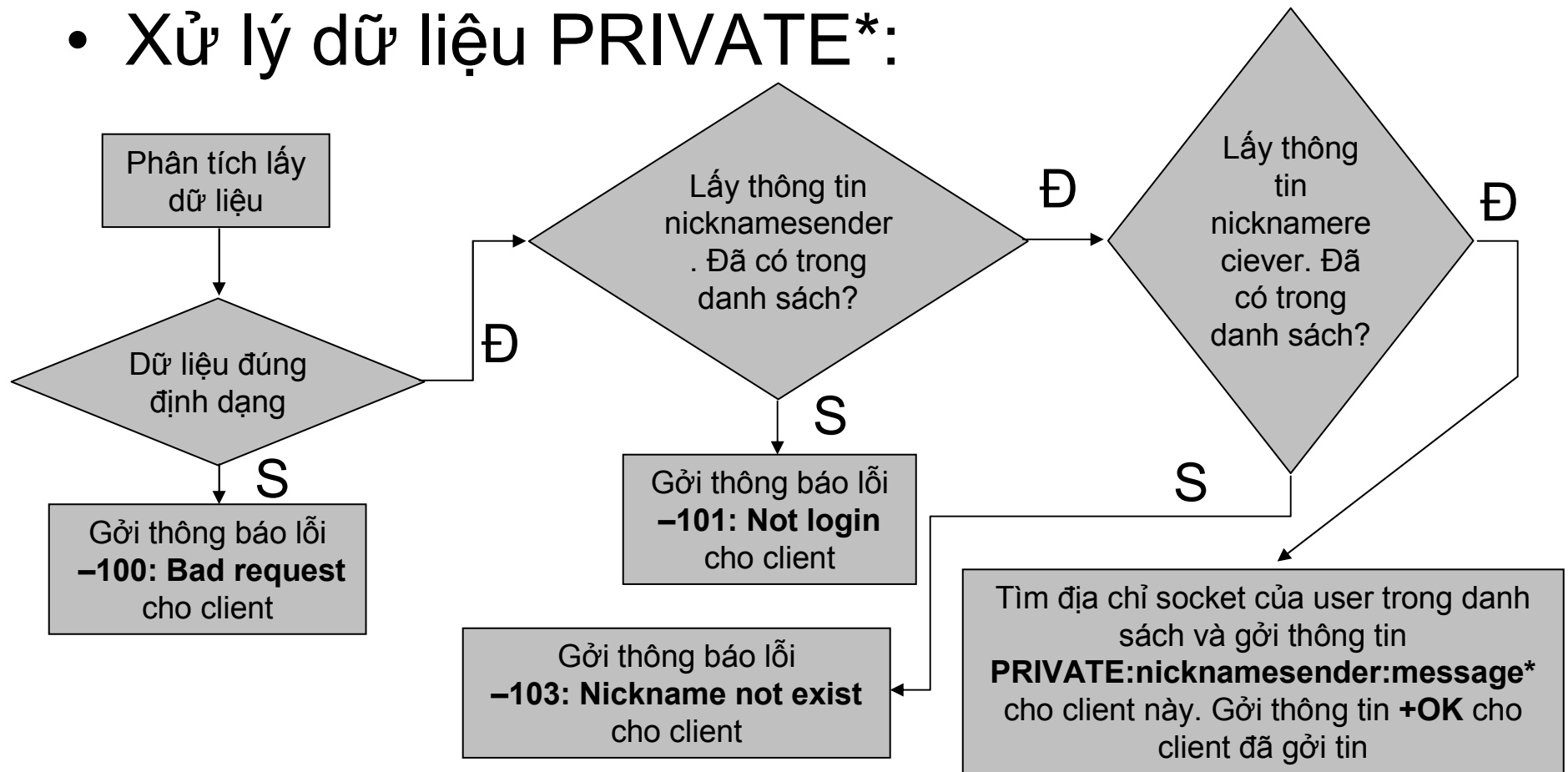
Thiết kế sơ đồ chức năng của ứng dụng MiniChatServer

- Xử lý dữ liệu PUBLIC*:



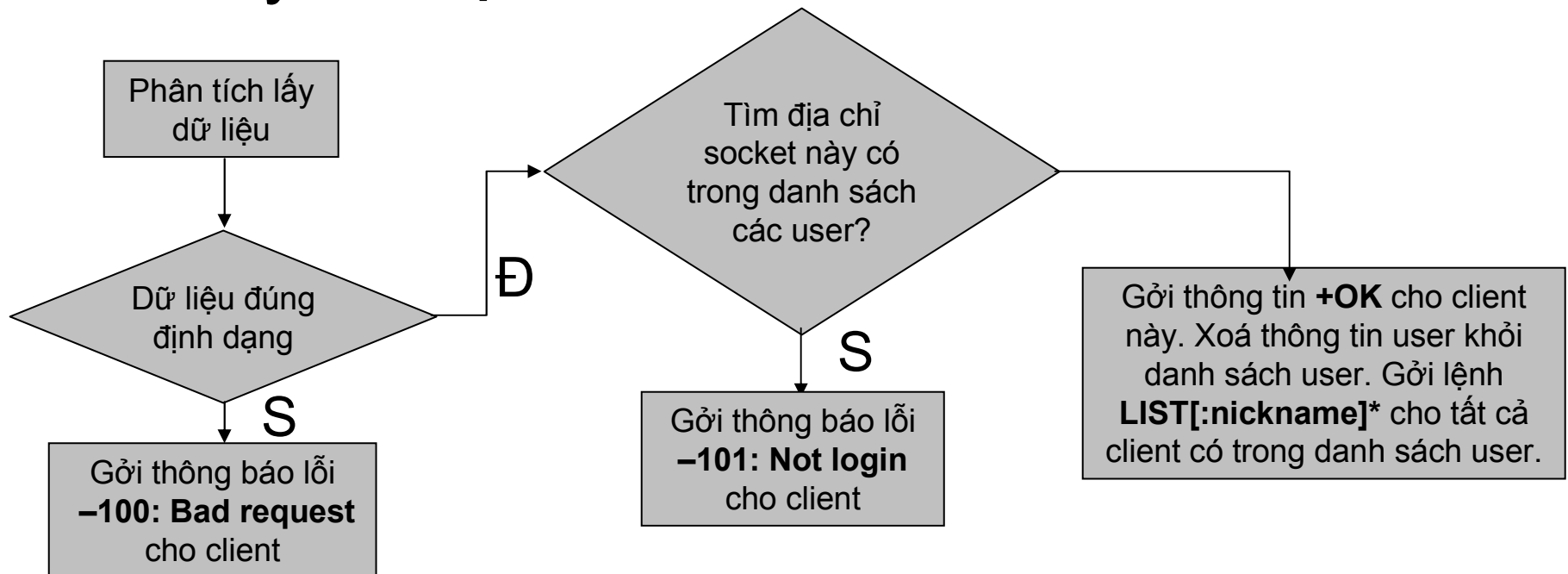
Thiết kế sơ đồ chức năng của ứng dụng MiniChatServer

- Xử lý dữ liệu PRIVATE*:



Thiết kế sơ đồ chức năng của ứng dụng MiniChatServer

- Xử lý dữ liệu QUIT*:



Thiết kế sơ đồ chức năng của ứng dụng MiniChatClient

- Chương trình **client** tạo socket, lấy các thông tin từ giao diện của người sử dụng để kết nối đến server.
 - Nếu kết nối thành công, lấy thông tin nickname để gửi dữ liệu LOGIN:nickname* đến server và chờ nhận dữ liệu về. Dữ liệu về có hai dạng:
 - nếu bắt đầu bằng ký hiệu '-' có nghĩa là bị lỗi, phân tích lỗi tương ứng để thông báo cho user
 - Nếu là LIST* có nghĩa là đăng nhập thành công, phân tích danh sách các nickname để hiển thị cho user
-

Thiết kế sơ đồ chức năng của ứng dụng MiniChatClient

- User gửi message vào chat room, có hai dạng:
 - Chỉ gửi cho một user: tạo lệnh PRIVATE* và gửi cho server. Chờ nhận dữ liệu về:
 - +OK: tiếp tục các quá trình khác
 - -xxx: lỗi giao thức, phân tích lỗi và thông báo
 - Gửi cho toàn bộ chat room: tạo lệnh PUBLIC* và gửi cho server. Chờ nhận dữ liệu về:
 - +OK: tiếp tục quá trình khác
 - -xxx: lỗi giao thức, phân tích lỗi và thông báo
-

Thiết kế sơ đồ chức năng của ứng dụng MiniChatClient

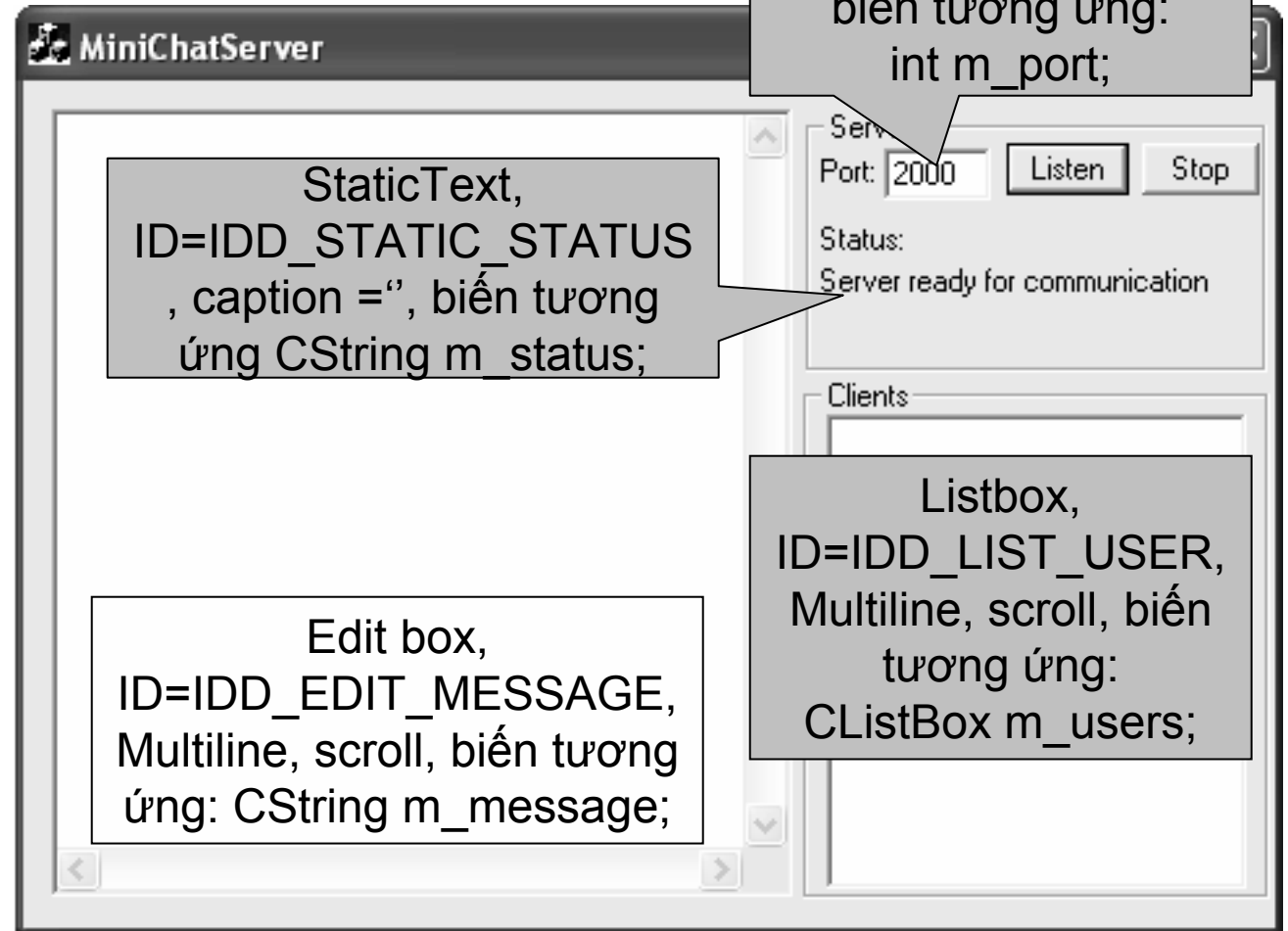
- User thoát khỏi chat room hoặc tắt chương trình
 - Gửi lệnh QUIT* cho server và chờ nhận dữ liệu về:
 - +OK: tiếp tục các quá trình khác
 - -xxx: lỗi giao thức, phân tích lỗi và thông báo
 - Nhận dữ liệu bất kỳ từ server: phân tích dữ liệu thuộc một trong các dạng: USERL*, USERQ*, PRIVATE*, PUBLIC*, nếu không thuộc các dạng này thì thông báo lỗi. Đối với mỗi dạng dữ liệu sẽ được xử lý như slide kế
-

Thiết kế sơ đồ chức năng của ứng dụng MiniChatClient

- Nếu là lệnh USERL*:
 - Phân tích lấy nickname
 - Thêm nickname vào danh sách các user
 - Nếu là lệnh USERQ*:
 - Phân tích lấy nickname
 - Xoá nickname khỏi danh sách các user, nếu có lỗi thì báo lỗi cho user
 - Nếu là lệnh PRIVATE* hoặc PUBLIC*
 - Hiển thị thông tin này cho user
-

Hiện thực chương trình MiniChatServer

Thiết kế giao diện như hình bên và đặt các biến tương ứng cho các phần tử giao diện



Hiện thực chương trình MiniChatServer

- Định nghĩa kiểu dữ liệu record để lưu danh sách các user (đầu file C*Dlg.h):

```
typedef struct T_UserRecord {  
    char name[20];  
    SOCKET socket;  
    int state;  
    struct T_UserRecord* next;  
} T_UserRecord;
```

- Khai báo các biến: (dùng chức năng Add Member Variable)

```
SOCKET ServerSocket; char temp_message[128];  
T_UserRecord *UserRecordList; T_UserRecord *tempUserRecord;
```

Hiện thực chương trình MiniChatServer

- Khai báo các hằng số (file Resource.h)
 - #define MSG_LENGTH 256
 - #define WSA_ACCEPT 1006
 - #define WSA_RDCLOSE 1007
 - #define CONNECTED 2000
 - #define LOGIN 2001
 - #define CHAT 2002
 - Lập trình theo các bước sau:
 - Tạo hàm xử lý sự kiện khi người dùng click chuột vào Button Listen, hàm OnButtonListen(): lần lượt gọi các lệnh socket, bind, listen, và WSASyncSelect
-

Hiện thực chương trình MiniChatServer

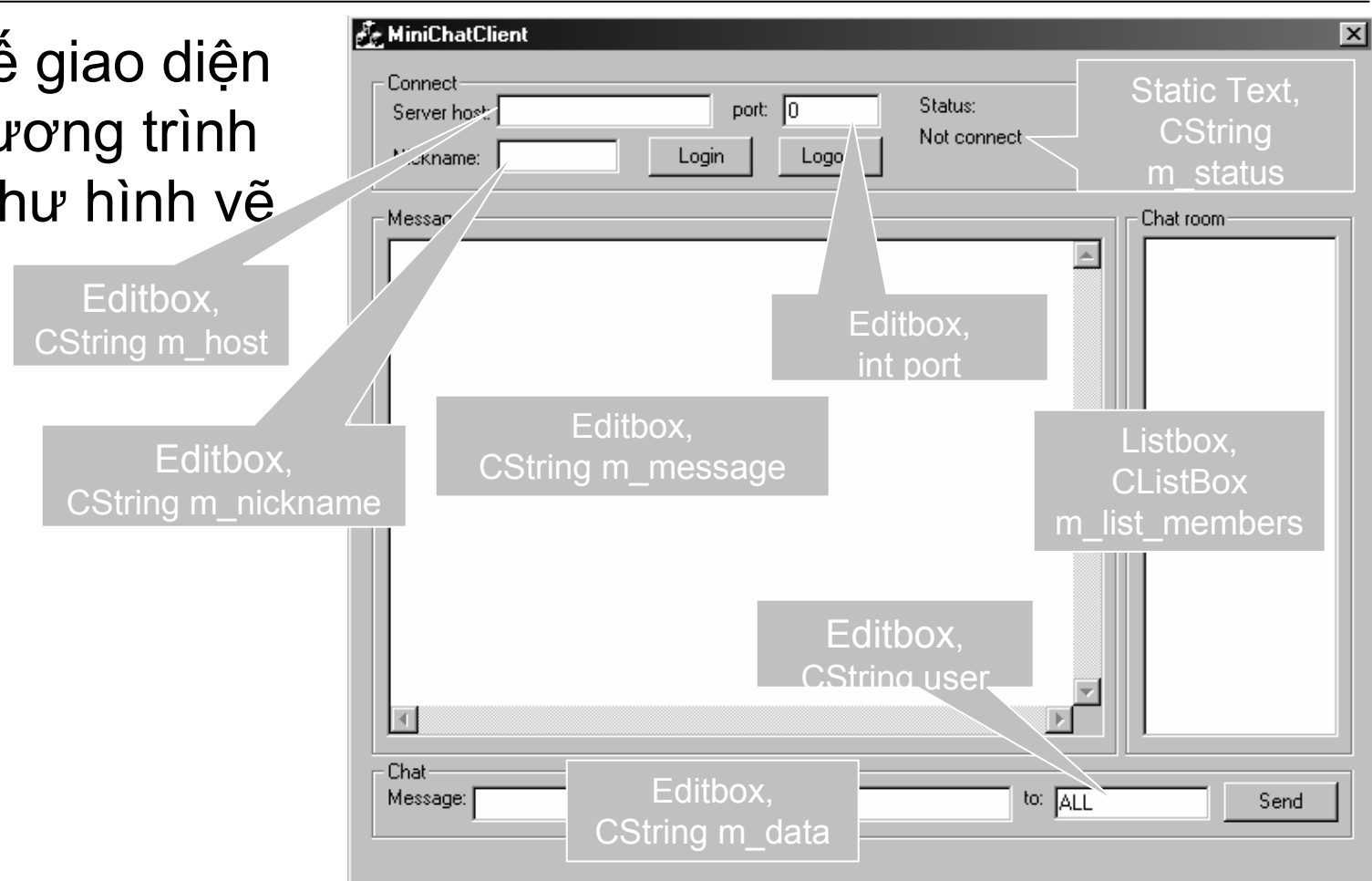
- Trình tự:
 - Tạo hàm WindowProc() và viết mã lệnh:
 - Tùy theo loại message sẽ gọi hàm tương ứng để xử lý (hai loại message định nghĩa là WSA_ACCEPT, WSA_RDCLOSE).
 - Định nghĩa và viết mã lệnh hàm OnAccept(...): xử lý sự kiện FD_ACCEPT, message WSA_ACCEPT khi có yêu cầu kết nối
 - Định nghĩa và viết mã lệnh hàm Process(...): xử lý sự kiện FD_READ, message WSA_RDCLOSE khi có dữ liệu gửi đến từ client
-

Hiện thực chương trình MiniChatServer

- Trình tự:
 - Trong hàm Process(): phân tích định dạng của dữ liệu đến xử lý tương ứng:
 - LOGIN*: hàm Login(...)
 - PUBLIC*: hàm SendPublic(...)
 - PRIVATE*: hàm SendPrivate(...)
 - QUIT*: hàm Logout(...)
 - Định nghĩa và viết mã lệnh cho lần lượt các hàm trên theo sơ đồ khối đã thiết kế
-

Hiện thực chương trình MiniChatClient

Thiết kế giao diện
của chương trình
client như hình vẽ



Hiện thực chương trình MiniChatClient

- Khai báo các biến: (dùng chức năng Add Member Variable)

```
SOCKET ClientSocket;  
char temp_message[128];  
int chat_status;  
CString data;
```

- Khai báo các hằng số (file Resource.h)

```
– #define NOTLOGIN          2000  
– #define LOGIN             2001  
– #define QUIT              2002  
– #define CHAT              2003  
– #define WSA_RDREAD        3000
```

Hiện thực chương trình MiniChatClient

- Lập trình theo các bước sau:
 - Tạo hàm xử lý sự kiện khi người dùng click chuột vào Button Login, hàm OnButtonLogin(): lần lượt gọi các lệnh socket, connect, và WSASyncSelect để chờ nhận sự kiện mạng
 - Tạo hàm WindowProc() và viết mã lệnh:
 - Chương trình client chỉ có một message (WSA_RDREAD) cho hai sự kiện FR_READ và FD_CLOSE, với mỗi sự kiện ta thực hiện lệnh gọi hàm tương ứng
 - Định nghĩa và viết mã lệnh hàm Process(...): xử lý sự kiện FD_READ, message WSA_RDREAD khi có dữ liệu từ server gửi đến
-

Hiện thực chương trình MiniChatClient

- Trình tự:
 - Định nghĩa và viết mã lệnh lần lượt các hàm Login, ResponseLogin, Send, Communicate, DisplayUserList
 - Tạo hàm xử lý sự kiện :OnSelchangeListMember, OnButtonSend, OnButtonLogout
-