**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

◆◆◆



**MÔN THỰC HÀNH CẤU TRÚC DỮ LIỆU**

**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

**Giảng viên thực hành: Nguyễn Vĩnh Tiệp**

**Lê Ngọc Thành**

**Nhóm:**

**Võ Anh Khoa 1512262**

**Nguyễn Văn Quang Huy 1512205**

**Nguyễn Quốc Huy 1512205**

**Trần Quang Thắng 1512525**

# Mở đầu:

## Giới thiệu – Phân công – Đánh giá các thành viên:

Nhóm:

Gồm 4 thành viên:

- Trưởng nhóm Võ Anh Khoa (1512262):

+ Chịu trách nhiệm level 3 và 4

+ Mức độ đóng góp trong nhóm: 30%

- Thành viên Nguyễn Văn Quang Huy (1512205):

+ Chịu trách nhiệm level 5 và viết báo cáo

+ Mức độ đóng góp trong nhóm: 30%

- Thành viên Nguyễn Quốc Huy (1512203):

+ Chịu trách nhiệm level 3 và 4

+ Mức độ đóng góp trong nhóm: 20%

- Thành viên Trần Quang Thắng (1512525):

+ Chịu trách nhiệm level 1 và 2

+ Mức độ đóng góp trong nhóm: 20%

## Kế hoạch:

Bước 1: Họp nhóm, phân chia công việc.

Bước 2: Thực hiện 3 công việc song song:

- Công việc 1:

+ Phân tích, thiết kế các lớp dùng để lưu trữ dữ liệu của kho hàng.

+ Tiến hành cài đặt level 1 và level 2.

- Công việc 2: Tìm hiểu về cách làm việc, cách sử dụng các thư viện đồ hoạ, cài đặt để tạo ra giao diện đồ hoạ GUI, phục vụ cho level 3 và 4.

- Công việc 3: Tìm hiểu về cách làm việc, xây dựng, cài đặt để có thể truyền và nhận dữ liệu giữa các máy với nhau, phục vụ cho level 5.

Bước 3:

- Tiếp tục hoàn thiện các lớp dùng để lưu trữ dữ liệu của kho hàng và xây dựng thêm một số lớp, hàm hỗ trợ khác

- Tiến hành cài đặt level 3 và 4

- Tiến hành cài đặt level 5.

Bước 4: Thực hiện 2 công việc song song:

- Công việc 1:

+ Ghép tất cả các phần của project lại với nhau

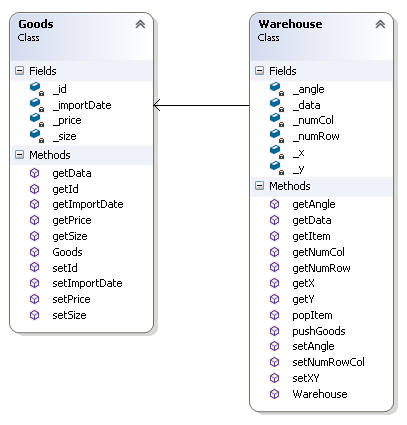
+ Chạy thử và kiểm tra

- Công việc 2: Viết báo cáo

# Báo cáo chi tiết:

## Phân tích, thiết kế các lớp dùng để lưu trữ dữ liệu của kho hàng:

- Sơ đồ UML:

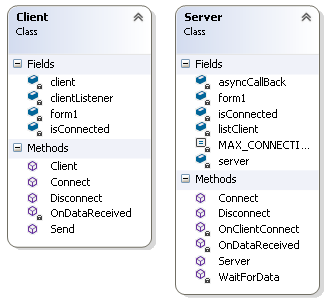


- Bảng chi tiết các lớp:

|  |  |  |
| --- | --- | --- |
|  | Lớp Goods | Lớp Warehouse |
| Mô tả | Lưu trữ dữ liệu của một kiện hàng trong nhà kho | Lưu trữ dữ liệu của một khu vực trong nhà kho |
| Thuộc tính | - \_id: string  Mã kiện hàng  - \_importDate: DateTime  Ngày nhập kho  - \_price: int  Đơn giá  - \_size: int  Kích thước | - \_angle: double  Góc nghiêng (đơn vị rad)  - \_data: Goods[,]  Mảng 2 chiều lưu giữ dữ liệu của tất cả kiện hàng trong khu vực  - \_numCol: int  Số cột  - numRow: int  Số dòng  - \_x: int  Toạ độ X  - \_y: int  Toạ độ Y |
| Phương thức | + getData(): string  Cung cấp thông tin của kiện hàng ở dạng chuỗi  + getId(): string  Cung cấp thông tin mã kiện hàng  + getImportDate(): DateTime  Cung cấp thông tin ngày nhập kho  + getPrice(): int  Cung cấp thông tin đơn giá  + getSize(): int  Cung cấp thông tin kích thước  + Goods()  Khởi tạo mặc định  + setId(string): Goods  Điều chỉnh thông tin mã kiện hàng  + setImportDate(string): Goods  Điều chỉnh thông tin ngày nhập kho  + setPrice(int): Goods  Điều chỉnh thông tin đơn giá  + setSize(int): Goods  Điều chỉnh thông tin kích thước | + getAngle(): double  Cung cấp thông tin góc nghiêng (đơn vị rad)  + getData(): string  Cung cấp thông tin của khu vực ở dạng chuỗi  + getItem(int,int): Goods  Cung cấp thông tin kiện hàng của khu vực ở dòng r, cột c  + getNumCol(): int  Cung cấp thông tin số cột  + getNumRow(): int  Cung cấp thông tin số dòng  + getX(): int  Cung cấp thông tin toạ độ X  + getY(): int  Cung cấp thông tin toạ độ Y  + popItem(int,int): Goods  Lấy kiện hàng ra khỏi khu vực ở dòng r, cột c  + pushItem(int,int,Goods): bool  Đưa kiện hàng item vào khu vực ở dòng r, cột c  + setAngle(double): Goods  Điều chỉnh thông tin góc nghiêng (đơn vị rad)  + setNumRowCol(int,int): Goods  Điều chỉnh thông tin số dòng và số cột. Toàn bộ thông tin các kiện hàng có trong kho trước đó sẽ bị xoá  + setXY(int,int): Goods  Điều chỉnh thông tin toạ độ X và Y  + Warehouse()  Khởi tạo mặc định |

- Một số lớp hỗ trợ:

Sơ đồ UML:



|  |  |  |
| --- | --- | --- |
|  | Lớp Client | Lớp Server |
| Mô tả | Đóng vai trò là người dùng, tương tác với database được đặt ở server | Đóng vai trò là server, quản lý database, là cầu nối trung gian kết nối tất cả người dùng lại với nhau, đảm bảo sự đồng bộ ở tất cả người dùng |
| Thuộc tính | - client: Socket  Socket được dùng để phục vụ kết nối giữa client với server  - clientListener: Thread  Thread con làm việc độc lập dùng để lắng nghe xem server có gửi dữ liệu xuống cho client không  - form1: Form1  Liên kết với form chính để thực hiện các thao tác xử lý dữ liệu ở client  - isConnected: bool  Trạng thái socket ở client có đang kết nối hay không | - asyncCallBack: AsyncCallBack  Dùng để xử lý bất đồng bộ khi các client gửi dữ liệu về không đồng bộ với nhau  - form1: Form1  Liên kết với form chính để thực hiện các thao tác xử lý dữ liệu ở server  - isConnected: bool  Trạng thái socket ở server có đang kết nối hay không  - listClient: List<Socket>  Danh sách các client đang kết nối với server  - MAX\_CONNECTION: int  Hằng số cho phép số lượng tối đa kết nối có thể có giữa client với server  - server: Socket  Socket được dùng để phục vụ kết nối giữa server với client |
| Phương thức | + Client(Form1)  Khởi tạo và kết nối với dữ liệu ở client  + Connect(string,string): bool  Kết nối tới server ở địa chỉ ip ipAddr ở cổng port  + Disconnect(): bool  Ngắt kết nối tới server  + OnDataReceive(): void  Xử lý khi có dữ liệu server gửi về  + Send(byte[]): bool  Gửi dữ liệu lên server | + Connect(string,string): bool  Mở kết nối ở địa chỉ ip ipAddr ở cổng port cho client có thể kết nối vào server  + Disconnect(): bool  Ngắt kết nối ở server  + OnClientConnect(IAsyncResult): void  Xử lý khi có client kết nối tới server  + OnDataReceive(IAsyncResult): void  Xử lý khi có dữ liệu client gửi về  + Server(Form1)  Khởi tạo và kết nối với dữ liệu ở server  + WaitForData(Socket): void  Chờ dữ liệu client gửi về |

- Một số hàm hỗ trợ:

+ SetData(StreamReader): List<Warehouse>

Đọc dữ liệu từ StreamReader và xuất ra danh sách thông tin các khu vực trong nhà kho

+ ModifyData(string): string

Đọc lệnh, kiểm tra xem lệnh có thực hiện được không. Nếu được thì thực hiện lệnh đó và chỉnh sửa lại dữ liệu nhà kho

+ ClientReceiveProcess(string): void

Khi client nhận được dữ liệu, câu lệnh này sẽ được gọi để thực hiện yêu cầu của server gửi xuống

+ ServerSendDatabase(): string

Khi client kết nối với server, câu lệnh này sẽ được gọi để thực hiện việc truyền database cho client để client bắt đầu phiên làm việc của mình

+ ServerReceiveProcess(string): string

Khi server nhận được dữ liệu, câu lệnh này sẽ được gọi để thực hiện yêu cầu của client và đưa ra phản hồi phù hợp

## Định dạng dữ liệu được lưu trữ trên bộ nhớ ngoài:

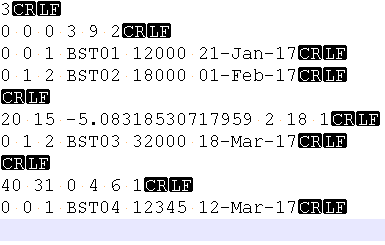
- Dòng đầu là số khu vực

- Các dòng tiếp theo là từng khu vực, mỗi khu vực cách nhau dòng trống

- Mỗi khu vực bắt đầu bằng thông tin mô tả về khu vực bao gồm: tọa độ X, tọa độ Y, góc nghiêng (radian), số dòng, số cột, số lượng kiện hàng đang có. Mỗi thông số cách nhau 1 khoảng trắng.

- Các dòng tiếp là thông tin của từng kiện hàng gồm: vị trí dòng, vị trí cột, kích thước, mã kiện hàng, đơn giá, ngày nhập kho. Mỗi thông số cách nhau 1 khoảng trắng.

VD:



Chú ý: Các ký tự khoảng trắng, xuống dòng,… cần phải được thể hiện 1 cách chặt chẽ như ở VD trên. Nếu không tuân thủ như vậy thì sẽ làm phá vỡ cấu trúc của file khiến file không thể đọc được.

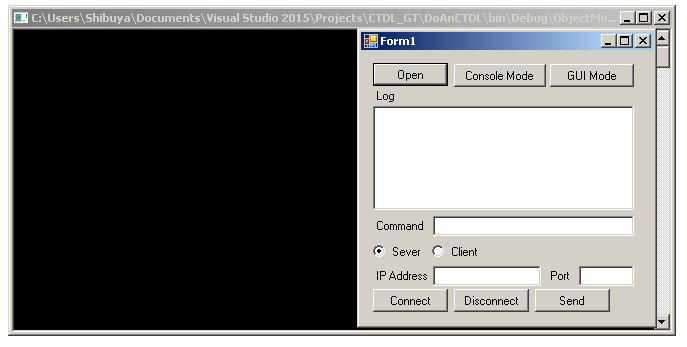
## Cài đặt level 1:

- Các hàm cài đặt cho level 1:

OpenButton\_Click: Khi nhấn button “Open” thì sẽ tạo hộp thoại cho phép chúng ta chọn file chứa dữ liệu của chương trình

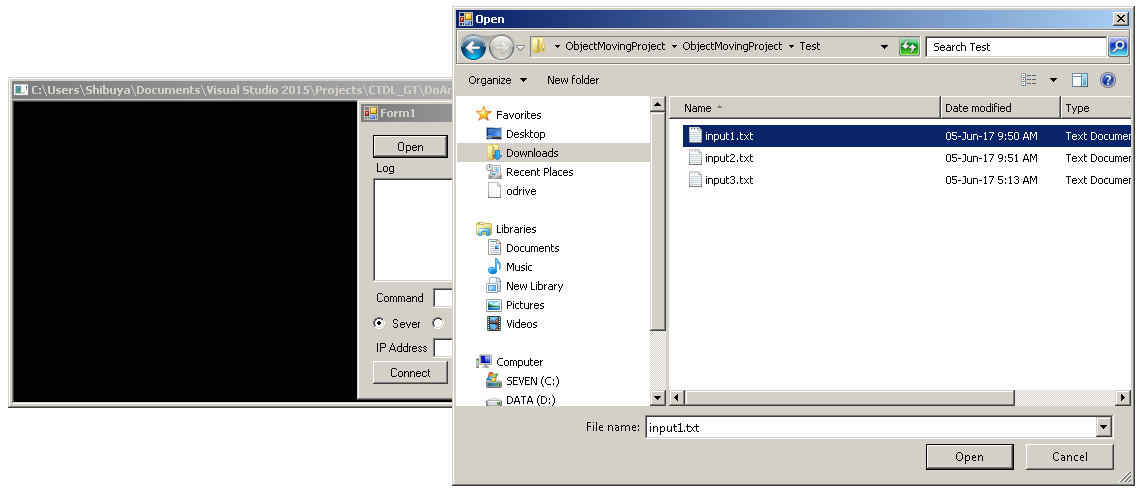
ConsoleButton\_Click: Khi nhấn button “Console Mode” thì sẽ chuyển qua chế độ làm việc trên console

- Giao diện chương trình:

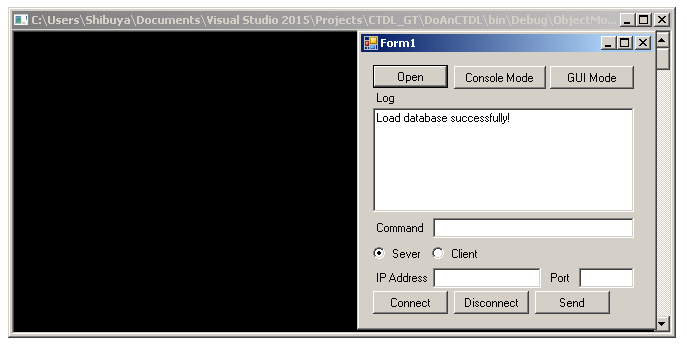


- Hướng dẫn sử dụng:

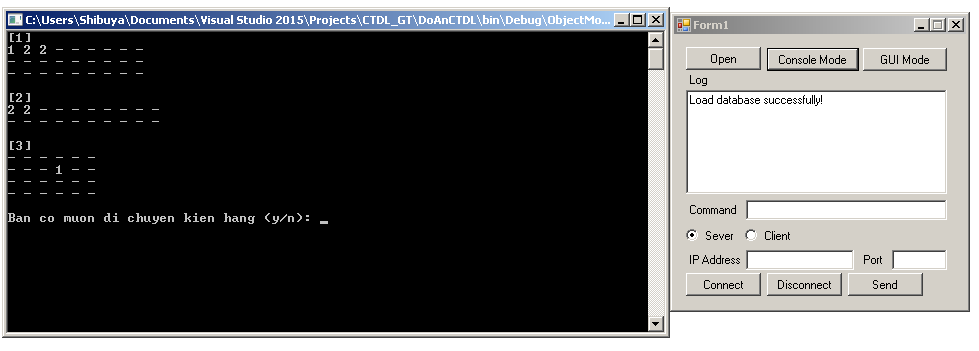
+ Nhấn button “Open” để tải dữ liệu vào:



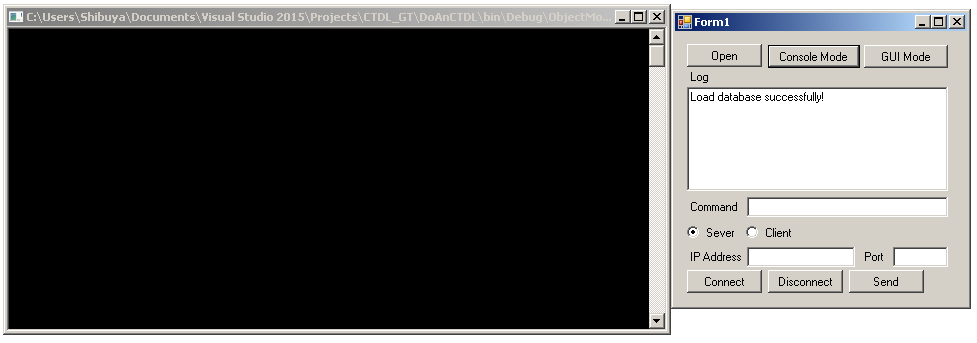
+ Thông báo sau khi load dữ liệu vào thành công:



+ Nhấn button “Console Mode” để bật chế độ console:



+ Gõ ký tự ‘n’ để thoát chế độ console:



## Cài đặt level 2:

- Các hàm cái đặt cho level 2:

ConsoleButton\_Click: Khi nhấn button “Console Mode” thì sẽ chuyển qua chế độ làm việc trên console

- Cú pháp lệnh:

Untitled.png

trong đó:

\_AreaFrom\_: là số nguyên, thể hiện khu vực có chứa kiện hàng cần di chuyển

\_XFrom\_, \_YFrom\_: là số nguyên, thể hiện toạ độ trong khu vực \_AreaFrom\_ có chứa kiện hàng cần di chuyển

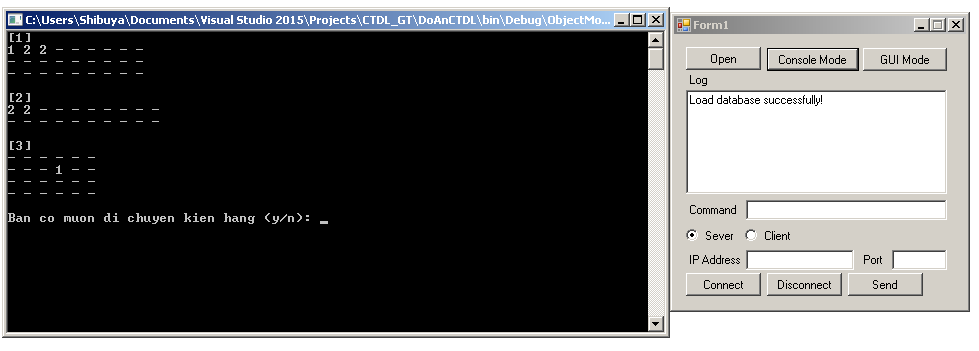
\_AreaTo\_: là số nguyên, thể hiện khu vực có chứa ô mà kiện hàng sẽ chuyển đến

\_XTo\_, \_YTo\_: là số nguyên, thể hiện toạ độ trong khu vực \_AreaTo\_ chứa ô mà kiện hàng sẽ chuyển đến

VD:

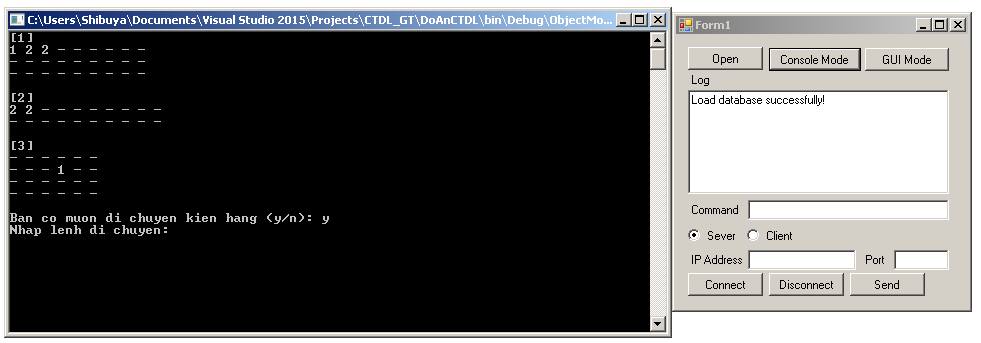
Untitled.png

- Giao diện chương trình:



- Hướng dẫn sử dụng: Sau khi load dữ liệu và bật chế độ console:

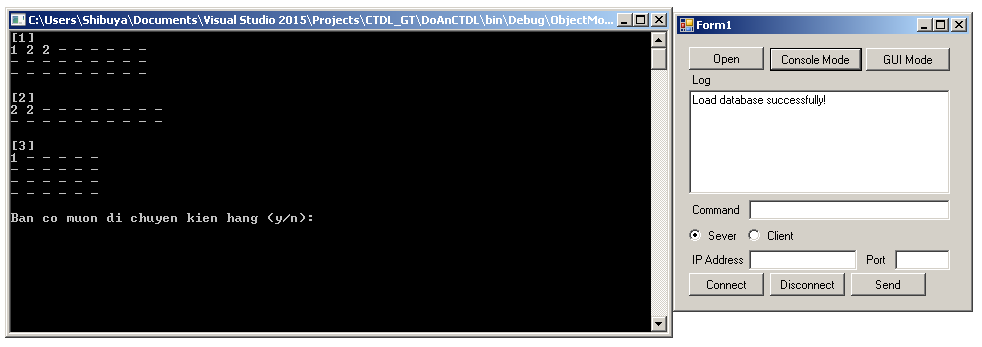
+ Gõ ký tự ‘y’ để nhập lệnh:



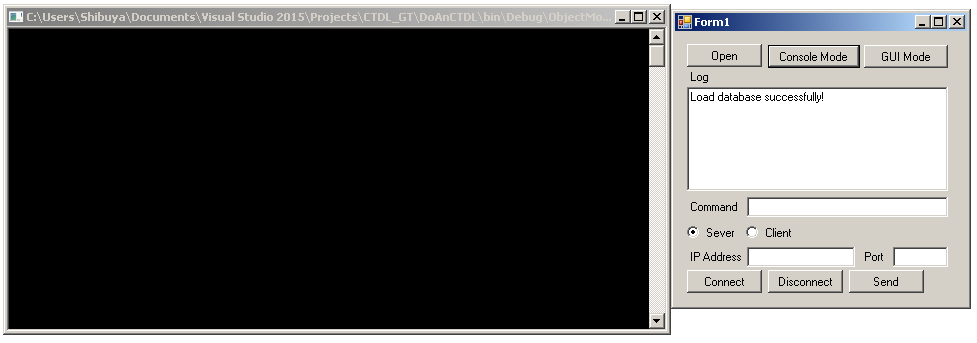
+ Gõ lệnh theo cú pháp. VD: M [3] (1,3) [3] (0,0)



+ Kết quả sau khi gõ lệnh:



+ Gõ ký tự ‘n’ để thoát chế độ console:



## Cài đặt level 3:

## Cài đặt level 4:

## Cài đặt level 5:

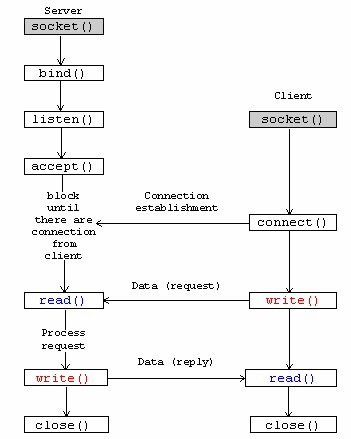
- Các hàm cái đặt cho level 5:

SendButton\_Click: Khi nhấn button “Send” thì client sẽ gửi dữ liệu lên server. Chú ý: hàm này chỉ sử dụng được khi client đã kết nối thành công tới server và hàm này chỉ dành cho chế độ server

ConnectButton\_Click: Khi nhấn button “Connect” thì đang ở trạng thái chưa hoặc không thiết lập kết nối nào cả sẽ chuyển qua trạng thái kết nối. Tuỳ vào chế độ mình chọn, kết nối này sẽ có dạng server hoặc client.

DisconnectButton\_Click: Khi nhấn button “Disconnect” thì chuyển từ trạng thái kết nối sang trạng thái ngắt kết nối.

- Mô hình thiết kế đằng sau 2 lớp client và server:



- Cú pháp lệnh:

IP Address: 4 số nguyên có kích thước 1 byte, ngăn cách nhau bởi dấu chấm.

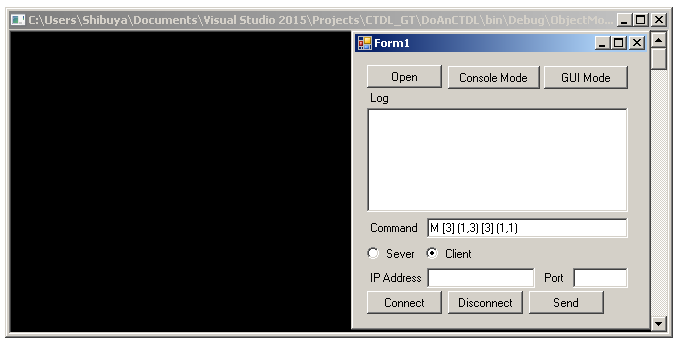
VD: 192.168.137.1

Port: là một số nguyên.

VD: 9090

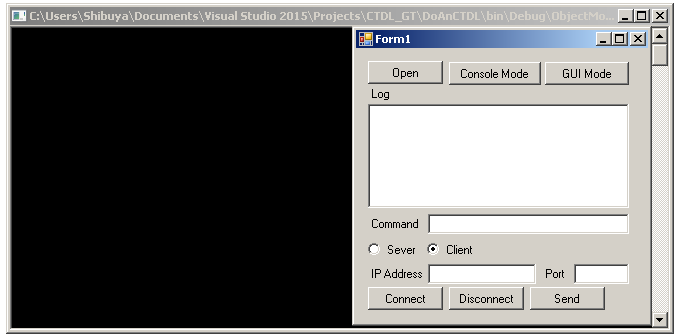
Command: là lệnh mà client gửi cho server. Cú pháp tương tự Cài đặt level 2. Nếu không theo cú pháp đó thì server sẽ hiểu ghi lại lệnh của client gửi đến trong log.

VD: M [3] (1,3) [3] (0,0)

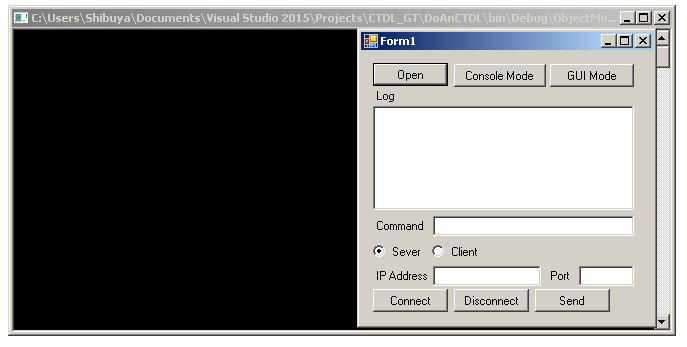


- Giao diện chương trình:

+ Client:



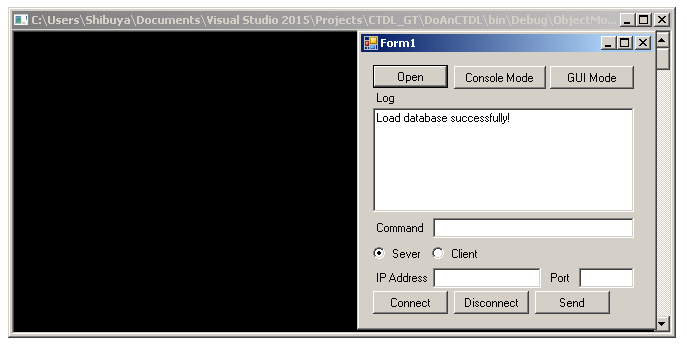
+ Server:



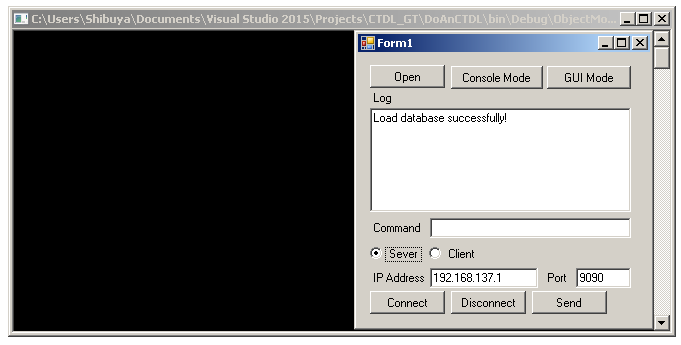
- Hướng dẫn sử dụng:

+ Ở Server:

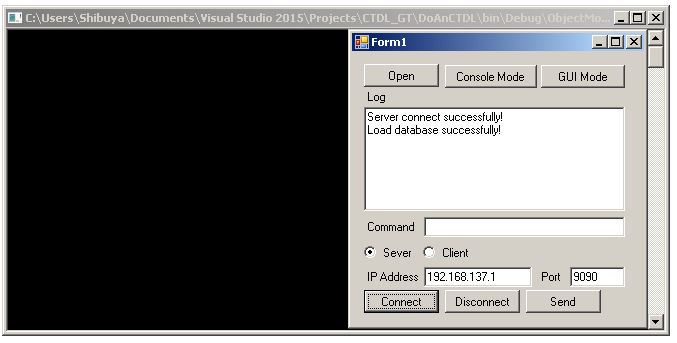
Nhấn button “Open” để tải dữ liệu vào:



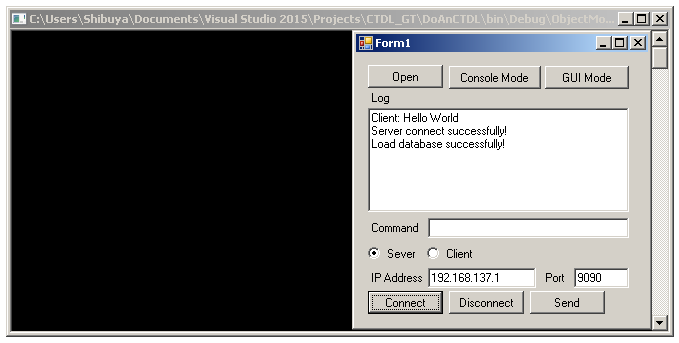
Chọn chế độ “Server”, đồng thời gõ địa chỉ IP và port dùng để mở cổng cho phép các client có thể kết nối tới Server:



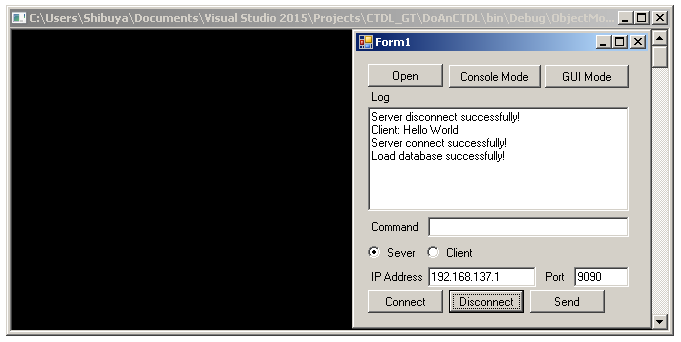
Nhấn button “Connect” để tiến hành kết nối:



Trong quá trình làm việc giữa server với client, server sẽ làm việc một cách tự động. VD khi client gửi “Hello World”, log của server xác nhận đã nhận được “Hello World” do client gửi:

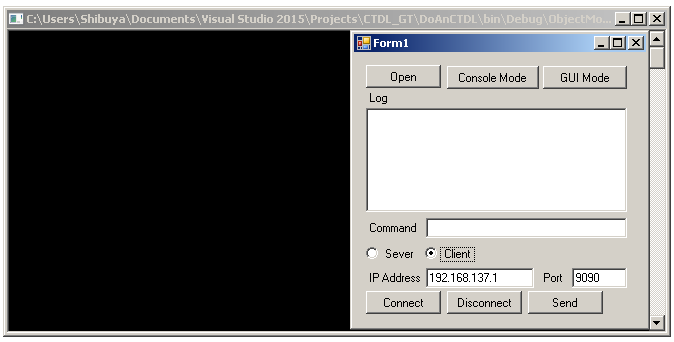


Nhấn button “Disconnect” để tiến hành ngắt kết nối:

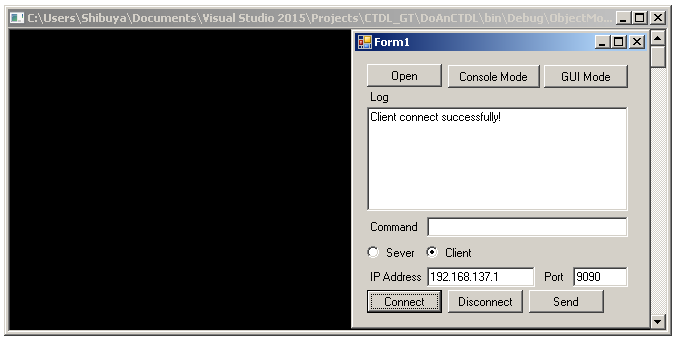


+ Ở Client:

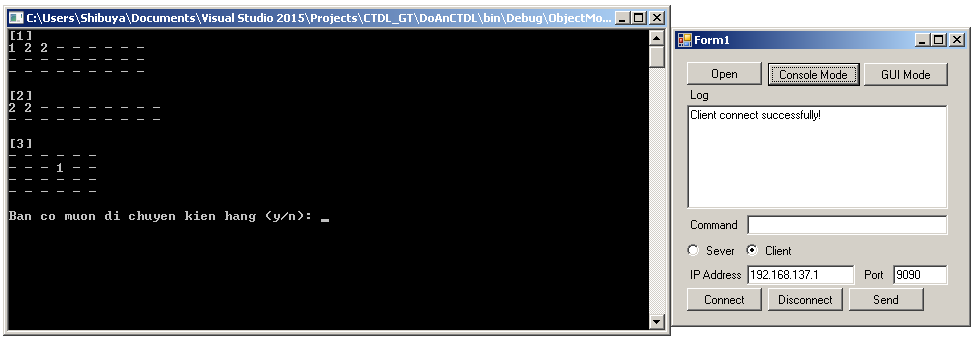
Chọn chế độ “Client”, đồng thời gõ địa chỉ IP và port của server mà client muốn kết nối tới:



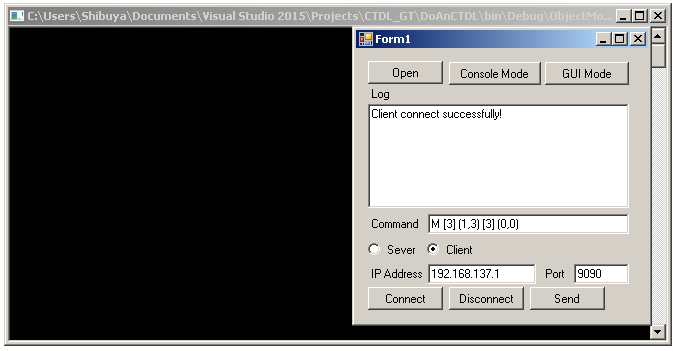
Nhấn button “Connect” để tiến hành kết nối:



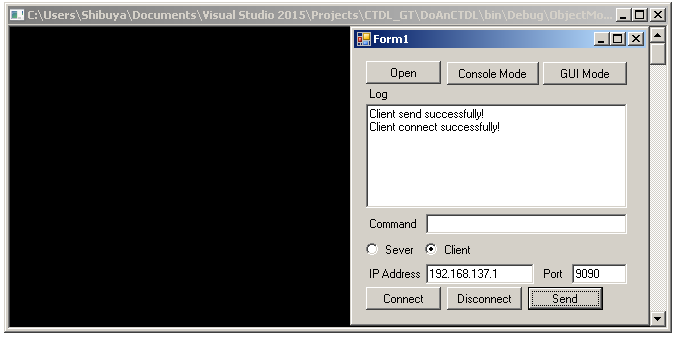
Kết quả sau khi client đã kết nối với server:



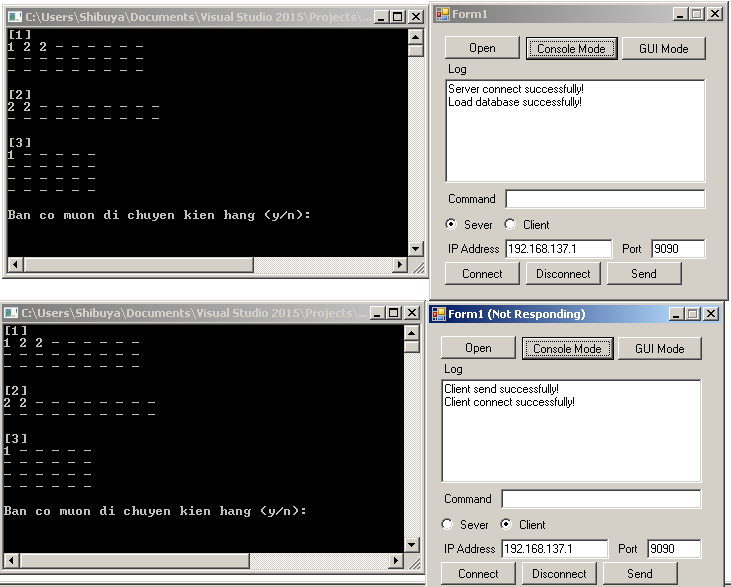
Gõ lệnh mà chúng ta muốn chương trình thực hiện:



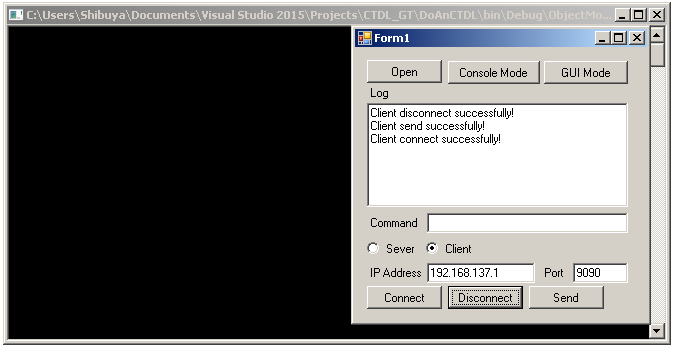
Nhấn button “Send” để tiến hành gửi lệnh:



Kết quả đạt được ở sever và client sau khi client gửi lệnh:



Nhấn button “Disconnect” để tiến hành ngắt kết nối:



Ghi chú: Khi chỉnh sửa kho hàng ở chế độ console và chế độ GUI thì client cũng sẽ gửi lệnh tương tự như vậy và như vậy mọi client sẽ được đồng bộ theo thời gian thực.

# Tổng kết:

Đã hoàn thành: 5/5 level