**MẠNG MÁY TÍNH**

**ENGLISH GAME**

PHÙNG THỊ HÒA - 1512186

C:\Users\tdqua_000\Dropbox\SS-Slides\DeCuong-CDIO\Template CDIO v4.2\Templates\Hinh anh\LogoTruong.png

Bộ môn Mạng máy tính

Khoa Công nghệ thông tin

Đại học Khoa học tự nhiên TP HCM

Mục lục

[1 Thông tin sinh viên 1](#_Toc498092413)

[2 Phát biểu bài toán sơ lược 1](#_Toc498092414)

[3 Cài đặt chương trình. 2](#_Toc498092415)

[4 Giao thức trao đổi giữa client và server. 5](#_Toc498092416)

[5 Kịch bản giao tiếp của chương trình 9](#_Toc498092417)

[6 Môi trường lập trình 11](#_Toc498092418)

[7 Đánh giá 11](#_Toc498092419)

[8 Nguồn tài liệu tham khảo 12](#_Toc498092420)

# Thông tin sinh viên

|  |  |
| --- | --- |
| **Họ và tên** | **Phùng Thị Hòa** |
| MSSV | 1512186 |
| Lớp | 15CTT1 |

# Phát biểu bài toán sơ lược

Lập trình Socket và xây dựng giao thức trao đổi cụ thể giữa client và server để viết trò chơi “English game”- là một phần trong các bài thi anh văn. Server cung cấp những câu tiếng anh chứa từ hoặc cụm từ sai về ngữ pháp, từ vựng. Người chơi có nhiệm vụ tìm ra lỗi sai trong câu và trả kết quả về cho server.

# Cài đặt chương trình.

#### Cơ chế lập trình non-blocking sử dụng WSAAsyncSelect

Trong lập trình Socket, nhiều hàm hoạt động theo cơ chế blocking (ví dụ: accept, recv). Khi gọi hàm accept(), chương trình server sẽ treo tới khi có một client gọi hàm connect() để kết nối đến. Trong thực tế, chúng ta ít khi nào chấp nhận để chương trình treo do đó phải sử dụng cơ chế non-blocking. Và để sử dụng non-blocking, chúng ta sử dụng WSAAsyncSelect.

Vậy WSAAsyncSelect là gì?

WSAAsyncSelect sử dụng cơ chế xử lý thông điệp của Windows. Khi windows kiểm tra và nhận thấy có một kết nối đến server sẽ gửi một thông điệp để báo cho server biết, lúc này server mới gọi hàm accept và không phải chờ đợi dẫn đến bị treo chương trình. Tương tự, khi có dữ liệu gửi đến, windows sẽ báo cho ứng dụng. Lúc này ứng dụng mới gọi hàm recv và nhận ngay dữ liệu.

Một số event sử dụng cài đặt trong bài là:

* FD\_READ: Báo hiệu có dữ liệu gửi đến cho socket, sẵn sàng để gọi hàm recv.
* FD\_ACCEPT: Báo hiệu cho soket server (đã gọi hàm listen) có một kết nối từ client.
* FD\_CLOSE: Báo hiệu đầu còn lại của kết nối đã đóng socket.

#### Các hàm được cài đặt chung ở cả client và server

* LRESULT SockMsg(WPARAM wParam, LPARAM lParam) là hàm xử lý các thông điệp của socket. Có sự khác nhau ở SockMsg Client và SockMsg Server diễn ra ở sự kiên FD\_READ vì cách xử lí thông điệp ở client và server khác nhau.
* char\* ConvertToChar(const CString &s) : đổi từ Cstring sang char\*.
* void Split(CString src,CString des[2]): tiến hành cắt thông điệp.
* void mSend(SOCKET sk, CString Command): gửi gói tin.
* int mRecv(SOCKET sk, CString &Command): nhận gói tin.

#### Cài đặt SockMsg ở Server phần sự kiên FD\_READ

Trong socketServer ta tạo cấu trúc:

struct Player

{

SOCKET sockClient;

char Name[200]; // tên sockCLient

int scores = 3; // Số điểm của sockClient

int limit = 0; // Sử dụng để kiểm tra client trả lời sai 3 câu liên tiếp hay không

int limitQuestion = 1; // Số câu hỏi đã trả lời

};

Trước tiên sử dụng biến chuỗi temp để lưu thông điệp được nhận.

Khi có nhiều client tham gia, thì công việc đầu tiên của chúng ta là xác định client đó là socketClient nào được lưu trong mảng của Server. Sử dụng biến post để lưu vị trí của client đó trong server.

Tiếp đó tiến hành sử dụng Split(temp, strResult) để chia lưu dòng thứ nhất của temp vào strResult[0], dòng thứ 2 lưu vào strResult[1] để tiếp tục thao tác.

Đổi strResult[0] thành số nguyên lưu vào flag.

Thực hiện xử lý:

-Nếu flag<0 thì xử lý các thao tác đăng kí, logout,start

+ flag=-1 xử lý login

+ flag=-2 xử lý logout

+ flag=-3 xử lý start

-Ngược lại thì xử lý theo các câu hỏi.

+ Trả lời đúng thì cộng điểm, limit=0. Tiến hành kiểm tra xem trả lời đủ 5

câu chưa. Nếu chưa đủ thì tiếp tục gởi câu hỏi.

+ Trả lời sai thì trừ điểm, limit++. Tiến hành kiểm tra đã trả lời đủ 5 câu

chưa? Số điểm có nhỏ hơn 0 không? Có bị sai ba lần liên tiếp không? Nếu

không có gì xảy ra thì tiếp tục gởi câu hỏi.

#### Cài đặt SockMsg ở Client phần sự kiện FD\_READ

Trước tiên sử dụng biến chuỗi temp để lưu thông điệp được nhận.

Tiếp đó tiến hành sử dụng Split(temp, strResult) để chia lưu dòng thứ nhất của strResult[0], dòng thứ 2 lưu vào strResult[1] để tiếp tục thao tác.

Đổi strResult[0] thành số nguyên lưu vào flag1.

Đổi strResult[1] thành số nguyên lưu vào flag2.

switch(flag1)

{

case -1:

if flag2=1 thì đăng kí thành công

else bị trùng tên đăng kí

case -4:

trả lời sai quá 3 câu

case -5:

bạn đã hoàn thành lượt chơi

case -6:

số điểm của bạn nhỏ hơn 0

default:

câu hỏi=strResult[1]

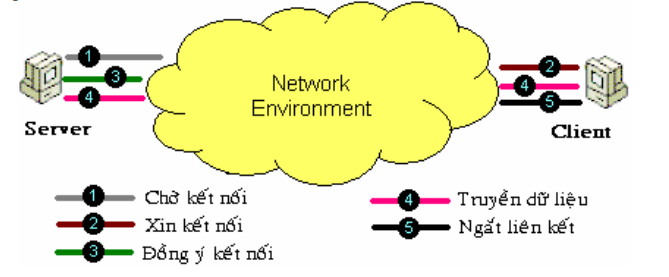
}

# Giao thức trao đổi giữa client và server.

#### Giao thức truyền thông

Sử dụng giao thức TCP (Transmistion Control Protocol) để thực hiện truyền nhận dữ liệu. TCP là một trong các giao thực cốt lõi của bộ giao thức TCP/ IP. Sử dụng TCP, các ứng dụng kết nối vơi nhau, qua đó chúng có thể trao đổi dữ liệu hoạc các gọi tin. Giao thực này đảm bảo được chuyển dữ liệu tới nơi nhận một các đáng tin cậy và có thứ tự. Ngoài ra, TCP còn phân biệt giữ dữ liệu của nhiều ứng dụng (ví dụ như dịch vụ Web và dịch vụ thư điện tử) đồng thời chạy trên một máy chủ. Với “English game” vấn đề cần đảm bảo ở đây là câu hỏi từ server được gởi tới client đầy đủ và chính xác, độ chính xác của truyền nhận dữ liệu đánh tin cậy. Với lí do đó mà giao thức được chọn sử dụng để truyền nhận dữ liệu giữa client và server là TCP.

#### Thứ tự truyền nhận thông điệp



Diễn tả thứ tự truyền nhận thông điệp bằng lời như sau: Server và Client đầu tiên sẽ được khai báo và khởi tạo. Server ở chế độ chờ nhận kết nối(gọi hàm listen). Client muốn kết nối sẽ gửi thông điệp đến server. Server sẽ chuẩn bị socket kết nối và tiếp đó sẽ đồng ý kiến nối với client (gọi hàm accept). Từ đây, Server và Client sẽ luân phiên thực hiện quá trình truyền nhận dữ liệu. Và sau cùng là đóng kết nối.

#### Kịch bản trao đổi giữa client và server

1. Server chờ nhận kết nối (listen).
2. Client gửi yêu cầu kết nối tới server (connect) và username dùng để đăng kí với server.
3. Server chấp nhận yêu cầu kết nối của client (accept), kiểm tra tính hợp lệ username của client đăng kí. Nếu username không hợp lệ (bị trùng với một username khác đã đăng kí) thì server gửi thông điệp để client biết username dùng để đăng kí không hợp lệ và để client thay đổi username. Quá trình này diễn ra đến khi nào username hợp lệ và được server chấp nhận.
4. Client gửi thông điệp với ý nghĩa đã đọc bảng hướng dẫn và sẵng sàng cho cuộc chơi.
5. Server nhận tín hiệu sẵn sàng chơi từ client, và gửi gói tin chứa câu hỏi đến cho client.
6. Client nhận gói tin chứa câu hỏi, tiến hành xử lý đề hiển thị câu hỏi và nhập câu trả lời. Client tiếp đó, gửi đi gói câu trả lời tới server để được xử lý.
7. Server nhận được gói tin trả lời từ client, sẽ kiểm tra gói tin vừa nhận, xử lý câu trả lời.
   1. Nếu trả lời sai liên tiếp 3 lần hoặc số điểm bé hơn 0 thì server gửi gói tin để báo cho client biết client phải dừng lượt chơi và ID sẽ bị xóa khỏi server.
   2. Nếu đã hoàn thành 5 câu hỏi của server thì server gởi gói tin để thông báo điểm cho người chơi.
   3. Nếu không rơi vào các trường hợp trên, server sẽ gửi gói câu hỏi tiếp theo đến client.

(Quá trình truyền nhận dữ liệu giữ client và server diễn ra tuần tự như vậy).

1. Client nhận gói tin của server, xử lý gói tin để đưa ra thông báo cho người chơi.
2. Client gửi tín hiệu đóng kết nối.
3. Server nhận tín hiệu đóng kết nối và ngắt kết nối với client.

#### Các thông điệp và ý nghĩa từng thông điệp

* SERVER gửi đi các thông điệp.

|  |  |
| --- | --- |
| **Thông điệp** | **Ý nghĩa** |
| listen(sockServer, 5); | Server chờ nhận kết nối. |
| accept(wParam,NULL,NULL); | Server chấp nhận kết nối từ client đã gởi yêu cầu kết nối. |
| \_T("-1\r\n1\r\n") | Thông báo cho client biết client đã đăng kí thành công. |
| \_T("-1\r\n2\r\n") | Thông báo cho client là tên đăng kí đã được sử dụng. |
| Num+\_T("\r\n")+question[num]+\_T("\r\n")  Trong đó:  + num: số thứ tự câu hỏi từ 0 đến 49.  + question[num]: nội dung câu hỏi thứ num trong danh sách câu hỏi. | Gởi gói tin câu hỏi đến người chơi. |
| \_T("-5\r\n") + diem + \_T("\r\n")  Trong đó: diem là số điểm hiển tại của người chơi. | Gởi số điểm người chơi đạt được. |
| \_T("-4\r\n\r\n") | Thông báo cho client biết client đã trả lời sai ba lần liên tiếp. |
| \_T("-6\r\n\r\n") | Thông báo cho client biết hiện tại số điểm của client <0. |

* CLIENT gửi thông điệp đi.

|  |  |
| --- | --- |
| **Thông điệp** | **Ý nghĩa** |
| Gọi hàm **connect** | Gởi thông điệp mong muốn kết nối. |
| \_T("-1\r\n")+ userName + \_T("\r\n")  Trong đó: userName ứng với tên người dùng muốn đăng kí. | Gởi yêu cầu đăng kí với server với tên là userName. |
| \_T("-2\r\n\r\n") | Gởi yêu cầu logout ngắt kết nối. |
| \_T("-3\r\n\r\n") | Gởi thông báo với server là có thể bắt đầu tham gia trò chơi. |
| num + \_T("\r\n") + answer + \_T("\r\n")  Trong đó: num là số thứ câu hỏi và answer là câu trả lời tương ứng với câu hỏi. | Gửi câu số thứ tự và câu trả lời tương ứng với câu hỏi. |

* Nhận xét chung về các thông điệp:
  + Tất cả các thông điệp để có dạng text như sau: a\r\nb\r\n

Với \r\n là ký tự xuống dòng.

* + “a” giả sử a đổi sang số nguyên là flag1.

flag1<0 để chúng ta biết sẽ phải xử lý các yêu cầu như registration, logout, start.

flag1>0 để biết được phải xử lí những câu trả lời.

* + Với flag1<0 ta tiến hành đổi “b” sang số nguyên flag2.

Ví dụ:

+ (flag1, flag2) =(-1,1): đăng kí thành công

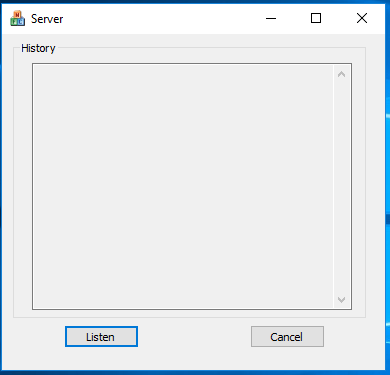
+ (flag1, flag2) =(-1,2): đăng kí thất bại (bị trùng user name).

* + Với flag1>0 ta lưu chuỗi question= b đối với client, answer=b đối với client.

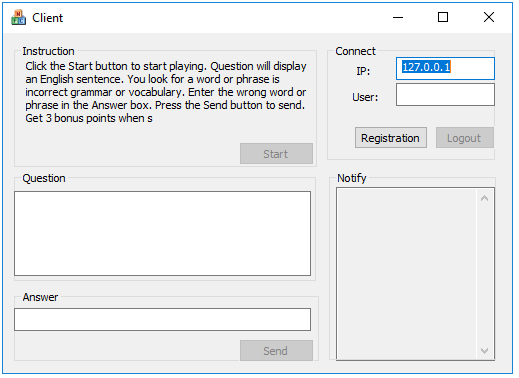
# Kịch bản giao tiếp của chương trình

#### Giao diện chính của chương trình

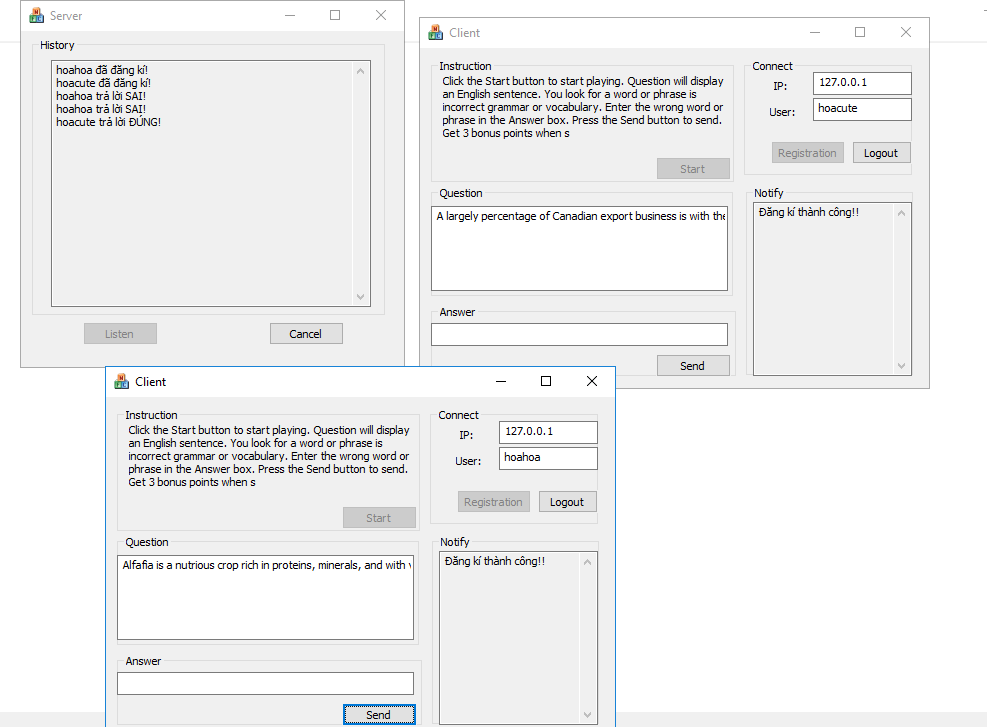
* Màn hình Server.



* Màn hình Client.



* Khi tham gia chơi nhiều người.



#### Kịch bản chính của chương trình

* Nhấn nút **Listen** ở server để bắt đầu lắng nghe các kết nối từ client.
* Các client nhập username vào ô **User**. Nhấn vào ô **Registration** để đăng kí, đăng kí thành công thì ở ô **Notify** sẽ hiển thị thông báo đã đăng kí thành công. Đọc xong hướng dẫn, nhấn **Start** để bắt đầu chơi. Đọc câu và tìm lỗi sai, nhập vào ô **Answer.** Nhấn  **Send**  để gửi câu trả lời. Sau khi trả lời 5 câu, không bị phạm quy, số điểm sẽ được thông báo ở Notify.

#### Luồng sự kiện phụ

* Nhấn nút **Listen** ở server để bắt đầu lắng nghe các kết nối từ client.
* Các client nhập username vào ô **User** , username sử dụng đã trùng với username đăng kí.
* Các client nhập username vào ô **User** , username vượt quá 8 ký tự. Messagebox hiển thị thông báo nhập lại tên.
* Khi client logout, Messagebox hỏi lại có muốn logout không?
* Nhập câu hỏi xong thay vì nhấn **Send**, nhấn enter chương trình sẽ tự động tắt.
* Các client nhập username vào ô **User**. Nhấn vào ô **Registration** để đăng kí, đăng kí thành công thì ở ô **Notify** sẽ hiển thị thông báo đã đăng kí thành công. Đọc xong hướng dẫn, nhấn **Start** để bắt đầu chơi. Đọc câu và tìm lỗi sai, nhập vào ô **Answer.** Nhấn  **Send**  để gửi câu trả lời. Trả lời sai ba câu liên tiếp hoặc số điểm <0 . Đưa ra thông báo ở **Notify** , người chơi phải dừng cuộc chơi và ID sẽ bị xóa khỏi chương trình.
* Các client nhập username vào ô **User**. Nhấn vào ô **Registration** để đăng kí, đăng kí thành công thì ở ô **Notify** sẽ hiển thị thông báo đã đăng kí thành công. Đọc xong hướng dẫn, nhấn **Start** để bắt đầu chơi. Đọc câu và tìm lỗi sai, nhập vào ô **Answer.** Nhấn  **Send**  để gửi câu trả lời. Chưa trả lời xong 5 câu, người chơi tự động ngắt chương trình, hay thao tác bấm nút bất kỳ thì ID người chơi bị xóa.

# Môi trường lập trình

#### Môi trường lập trình

Visual Stdio 2017

Ngôn ngữ lập trình C/C++

#### Các framework hỗ trợ thực thi

Thư viện máy tính: Microsoft Foundation Classes (MFC).

# Đánh giá

#### Các chức năng đã làm được.

* Nhiều người dùng có thể tham gia trò chơi cùng lúc.
* Người dùng được quyền đăng ký với server để tham gia trò chơi. Mỗi ID đăng ký không quá 8 ký tự, và không được đăng ký trùng với các user hiện tham gia trò chơi.
* Với mỗi lượt chơi:

+ Server gửi câu hỏi cho client sau khi đã chấp nhận ID client đăng ký. Thứ tự các câu hỏi được random từ bộ đề của server. (Bộ đề của server có ít nhất 30 câu hỏi)

+ Người dùng nhận 1 câu hỏi từ server, tìm ra từ sai trong câu.

+ Người dùng gửi đáp án về server.

+ Server trả kết quả về cho client: đáp án đúng: + 1 điểm, sai – 1 điểm. Số điểm ban đầu được tặng 3 điểm.

+Server thông báo kết quả cho client:

* Dừng cuộc chơi nếu số điểm <0 hoặc trả lời sai 3 lần liên tiếp. ID của client sẽ bị xóa khỏi server.
* Thông báo tổng số điểm: khi client trả lời hết số câu hỏi server đưa ra

+ Client không được phép dừng cuộc chơi khi chưa trả lời hết câu hỏi. Nếu client đơn phương dừng cuộc chơi (ngắt kết nối, tắt chương trình), server xóa ID của client.

#### Các chức năng chưa làm được.

+ Thời gian cho mỗi câu hỏi là 1 phút, sau 1 phút nếu server chưa nhận được câu trả lời từ client, xem như client mất điểm trong câu này.

#### Mức độ hoàn thành.

Hoàn thành 90% yêu cầu đề bài.

# Nguồn tài liệu tham khảo

Các file code demo của thầy.

Sách: Lập trình Windows với MFC mục “MFC với internet” của tác giả Lê Ngọc Thạch.