ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN ĐIỆN

[](https://www.google.com.vn/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwj_gYO6sPTeAhWZfCsKHdDgBnoQjRx6BAgBEAU&url=https://www.hust.edu.vn/brand-identity&psig=AOvVaw07rpUr1V_JYrwqlpPqMPlH&ust=1543400782595482)

BÁO CÁO ĐỒ ÁN

MÔN KĨ THUẬT LẬP TRÌNH

TRÒ CHƠI TÀU CHIẾN – BATTLESHIPS

Giảng viên: TS Nguyễn Việt Tùng

Nhóm sinh viên: MSSV

Nguyễn Xuân Khải 20173970

Lương Minh Kiên 20173989

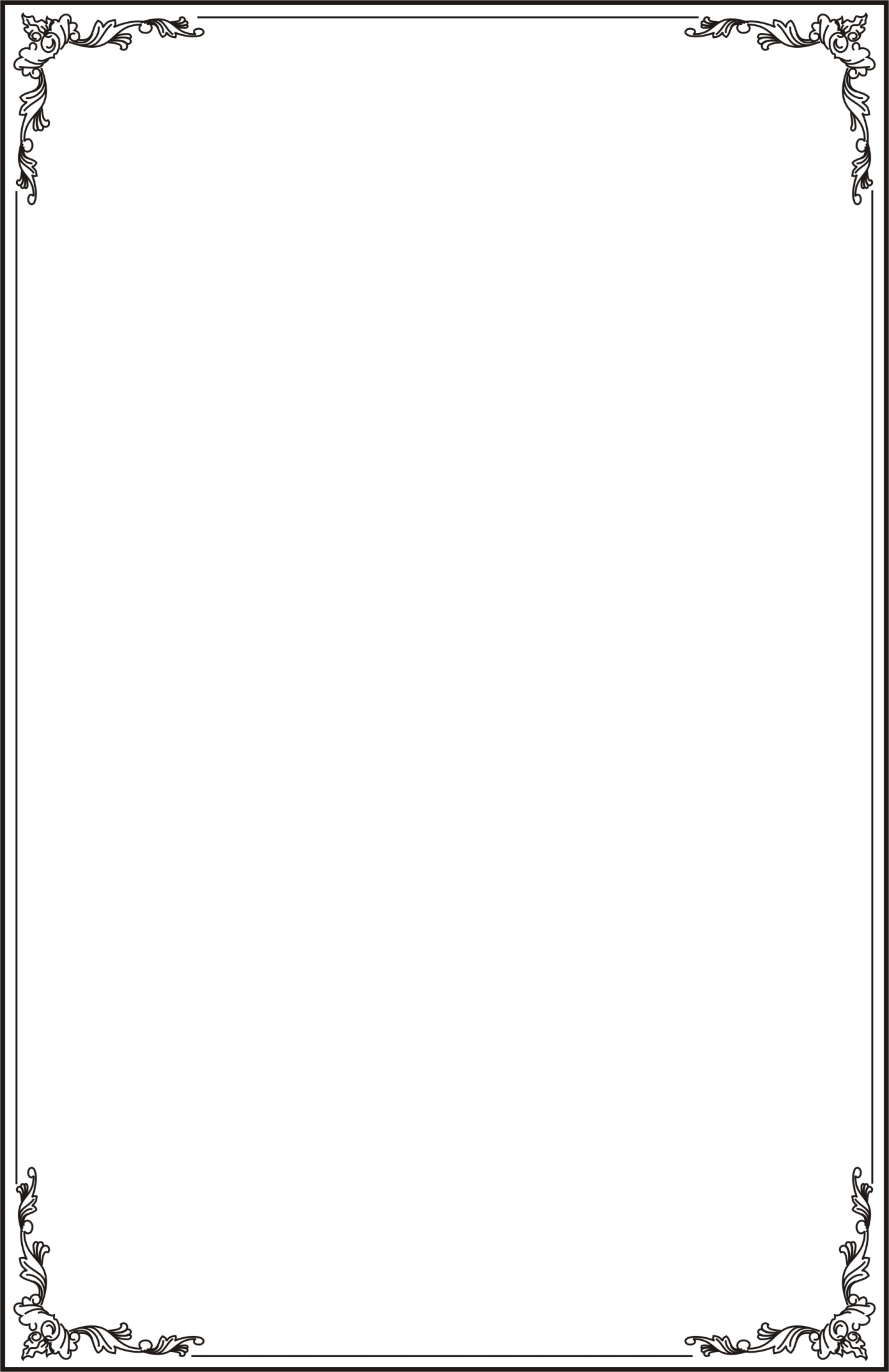
Lê Công Nhật Anh 20173649

Dương Văn Dũng 20173052

Nguyễn Trọng Khánh 20173983

Bùi Huy Đức 20173748

Lớp: CTTN- Điều Khiển Tự Động K62

Hà Nội, tháng 12 năm 2019

# NHẬN XÉT CỦA GIẢNG VIÊN

..................................................................................................................................................................................................................................................................................................................................................................................................................................................................... .............................................................................................................................................................................................................................................................................................................. .............................................................................................................................................................................................................................................................................................................. .............................................................................................................................................................................................................................................................................................................. .............................................................................................................................................................................................................................................................................................................. .............................................................................................................................................................................................................................................................................................................. .............................................................................................................................................................................................................................................................................................................. .............................................................................................................................................................................................................................................................................................................. .............................................................................................................................................................................................................................................................................................................. .............................................................................................................................................................................................................................................................................................................. .............................................................................................................................................................................................................................................................................................................. .............................................................................................................................................................................................................................................................................................................. .............................................................................................................................................................................................................................................................................................................. .............................................................................................................................................................................................................................................................................................................. .............................................................................................................................................................................................................................................................................................................. ..............................................................................................................................................................................................................................................................................................................

# MỤC LỤC

[LỜI MỞ ĐẦU 4](#_Toc26532302)

[CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN 5](#_Toc26532303)

[CHƯƠNG 2: ĐỊNH HƯỚNG VÀ PHÂN CÔNG BÀI TOÁN 7](#_Toc26532304)

[2.1. ĐỊNH HƯỚNG BÀI TOÁN 7](#_Toc26532305)

[2.2. PHÂN CHIA CÔNG VIỆC 8](#_Toc26532306)

[CHƯƠNG 3: THIẾT KẾ VÀ THỰC HIỆN CHƯƠNG TRÌNH 9](#_Toc26532307)

[3.1. CÁC CHẾ ĐỘ CHƠI 9](#_Toc26532308)

[3.2. CÁC THƯ VIỆN, HẰNG, BIẾN DỮ LIỆU 9](#_Toc26532309)

[3.2.1. Thư viện 9](#_Toc26532310)

[3.2.2. Hằng 9](#_Toc26532311)

[3.2.3. Biến dữ liệu 10](#_Toc26532312)

[3.3. CÁC HÀM SỬ DỤNG 10](#_Toc26532313)

[CHƯƠNG 4: CÀI ĐẶT CHƯƠNG TRÌNH 29](#_Toc26532314)

[4.1. ƯU ĐIỂM 29](#_Toc26532315)

[4.2. NHƯỢC ĐIỂM 29](#_Toc26532316)

[4.3. HƯỚNG PHÁT TRIỂN CỦA NHÓM 29](#_Toc26532317)

[TÀI LIỆU THAM KHẢO 30](#_Toc26532318)

# LỜI MỞ ĐẦU

Ngôn ngữ lập trình C++ được phát triển từ ngôn ngữ C. Nó chính là mở rộng của ngôn ngữ, có nghĩa là nó được bổ sung các đặc tính mới của ngôn ngữ lập trình hướng đối tượng. Trước đó ngôn ngữ C là ngôn ngữ bậc cao được biết đến với khả năng chạy được trên hầu hết các máy tính và độc lập với phần cứng. Ngôn ngữ lập trình C++ được phát triển bởi Bjarne Stroustrup vào đầu những năm 80 tại phòng thí nghiệm Bell. C++ so với ngôn ngữ lập trình C có 2 điểm mở rộng như bổ sung thêm tham chiếu, hàm chồng, tham số mặc định... và hỗ trợ lập trình hướng đối tượng. Với những ưu điểm này, nhóm em đã chọn C++ làm ngôn ngữ lập trình chính trong dự án của mình.

Mặt khác, nhu cầu giải trí tăng cao cùng với sự phát triển của các nền tảng công nghệ khiến cho phát triển game trở thành một lĩnh vực cực kỳ có sức hút đối với giới trẻ. Để phát triển được một trò chơi hoàn chỉnh cần nhiều yếu tố về con người và máy móc, trong đó không thể không kể đến yếu tố lập trình. Nhằm áp dụng các kiến thức đã học trong bộ môn Kĩ Thuật Lập Trình cũng như tự tìm hiểu thêm các khía cạnh mới, nhóm em đã quyết định chọn đề tài lập trình game Battleship – Tàu Chiến. Đây là một trò chơi đã có lịch sử lâu đời và mang tính chiến thuật cao.

Dù đã rất cố gắng, nhóm em cũng khó có thể tránh khỏi những thiếu sót nhất định. Rất mong nhận được sự đóng góp ý kiến của các Thầy Cô, để chương trình ngày càng hoàn thiện hơn.

Xin Chân Thành Cảm Ơn!

# CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN

Battleship là một game chiến thuật được chơi bởi hai người chơi đối kháng trên bản đồ vùng biển của mỗi người.

Mục tiêu :

Đánh chìm tất cả các của địch hoặc gây ra nhiều sát thương cho hạm đội hơn đối thủ khi cả hai cùng hết số tên lửa được cung cấp.

Thành phần:

* Hai bàn cờ
* Một số lượng tàu chiến nhất định

Chuẩn bị :

* Mỗi người chơi có một hạm đội tàu giống hệt nhau. Trong một hạm đội có đa dạng các loại tàu khác nhau.
* Trước khi bắt đầu giao chiến, mỗi người chơi bí mật đặt tàu chiến trong vùng biển của mình.
* Người chơi sẽ không biết được cách đặt tàu của đối phương như thế nào.
* Mỗi người được cung cấp số lượng tên lửa nhất định.

Luật đặt tàu chiến:

* Đặt tàu theo chiều ngang hoặc dọc tương ứng với các ô trên bản đồ, không được đặt chéo hoặc hình ziczắc.
* Không được đặt các tàu chồng lên nhau.
* Không được thay đổi vị trí của tàu khi trò chơi bắt đầu.

Cách chơi:

* Chọn một người chơi bắn trước .
* Bạn và đối phương chơi thay phiên nhau.
* Mỗi đợt bạn bắn 1 tên lửa cố bắn trúng tàu đối phương.

Giao chiến :

* Mỗi người chơi có hữu hạn số tên lửa.
* Trong lượt của mình, bạn chọn một vị trí trong vùng biển đối phương để bắn tên lửa vào.
* Mỗi vị trí trong bảng hiện ra trong lượt chơi của bạn tương ứng với vị trí trên vùng biển đối phương.

Bắn trượt và bắn trúng:

* Khi bạn chọn một vị trí để bắn, nếu bắn trúng một phần của một con tàu sẽ ngay lập tức đánh chìm tàu đó, hiện kí hiệu trúng trên bản đồ, nếu bắn trượt sẽ hiện kí hiệu trượt.
* Các vết bắn trúng tàu đối phương sẽ được lưu lại trên bản đồ.
* Các vết bắn trượt cũng được lưu lại trên bản đồ để bạn biết và không bắn vào đó nữa.
* Bất kể bắn trúng hay bắn trượt, lượt chơi của bạn đã kết thúc.
* Điểm số được tính bằng với kích thước của tàu mà bạn bắn hạ.
* Chơi tiếp tục như vậy, mỗi đợt bạn và đối phương bắn một lần.

Kết quả trò chơi:

* Người chiến thắng là người bắn chìm hết tàu đối phương.
* Trường hợp cả 2 người chơi đều không băn hết tàu đối phương khi đã hết tên lửa, người chơi gây ra nhiều sát thương hơn là người chiến thắng.

# CHƯƠNG 2: ĐỊNH HƯỚNG VÀ PHÂN CÔNG BÀI TOÁN

## **2.1. ĐỊNH HƯỚNG BÀI TOÁN**

Battleship là một game chiến thuật có thể được chơi bởi 2 người chơi. Mỗi người chơi sẽ bắt đầu với 1 bản đồ rỗng hình vuông được chia ô N x N (N > 3); một bản đồ rỗng với kích thước N x N có thể được lưu vào file theo định dạng như sau:

4

0 0 0 0

0 0 0 0

0 0 0 0

0 0 0 0

mapPlayer1.txt

Với dòng đầu tiên là N. N dòng tiếp theo, mỗi dòng có N số 0, cách nhau bởi một khoảng trắng.

Mỗi người chơi có một đội tàu giống hệt nhau bao gồm M tàu (M > 2). Mỗi tàu là một hình khối có kích thước 1 x d (1 <= d <= 3). Người chơi có thể đặt các tàu của mình vào vị trí bất kì trên bản đồ, vị trí có tàu sẽ được đánh số, vị trí không có tàu sẽ được đánh số 0 và lưu vào FILE “mapPlayer1.txt”. Mỗi tàu được đánh số khác nhau. Các tàu chỉ có thể nằm ngang hoặc dọc (không thể nằm chéo hoăc zic zắc). Như vậy, ở đây người chơi trực tiếp làm việc với FILE .TXT. Ví dụ:

4

0 1 0 0

3 0 0 0

3 0 2 2

3 0 0 0

mapPlayer1.txt

Mỗi người chơi có K (K < N\*N) tên lửa (đảm bảo người chơi không thể bắn hết tất cả các điểm trên bản đồ), mỗi tên lửa chỉ có thể bắn được vào 1 ô vuông nhất định trên bản đồ của đối phương. Trong đồ án này, nhóm em đặt K = N\*N/2. Người chơi lần lượt chơi bằng cách nhập tọa độ ô mong muốn bắn tên lửa đến. Sau mỗi lần nhập, chương trình in ra bản đồ của người bị bắn tên lửa đến với kết quả bắn trúng hay trượt từ lượt chơi đầu tiên cho đến hiện tại. Nếu bắn trúng, hiển thị ra X và nếu bắn trượt, hiển thị ra M.

Trò chơi kết thúc khi: 1 trong 2 người chơi bị bắn hết tàu hoặc cả 2 người chơi bắn hết số tên lửa của mình. In ra màn hình người chiến thắng (nếu bắn hết được tàu đối phương trước, nếu có số lần bắn trúng nhiều hơn) hoặc in ra màn hình nếu kết quả hòa.

Viết chương trình cài đặt trò chơi battleship mô tả bên trên với 2 chế độ chơi: người với người, máy với người. Trong đó, máy tính sẽ thực hiên các bước bắn tên lửa theo thuật toán nào đó hoặc ngẫu nhiên sao cho không có 2 lần bắn tên lửa nào bắn vào cùng 1 ô.

Người chơi sẽ được lần lượt yêu cầu nhập vào:

* N để khởi tạo bản đồ cho cả 2.
* M là số lượng tàu cho cả 2, kích thước của từng tàu (1 x d).
* Bản đồ đặt tàu cho từng người chơi được lưu vào 2 file mapPlayer1.txt và mapPlayer2.txt

Yêu cầu nâng cao: (tùy chọn, điểm thưởng) Mỗi tàu của người chơi sẽ có một giá trị riêng biệt tương ứng với chiều dài d của tàu (1<=d<=3). Tàu bị bắn có giá trị cao sẽ đem về nhiều điểm hơn.

Ví dụ dưới đây thể hiện việc đặt vào bản đồ 04 tàu, có độ dài và giá trị khác nhau:

0 2 2 0

3 0 1 0

3 0 0 2

3 0 0 2

Mức độ đáp ứng yêu cầu:

* Các vấn đề gốc của trò chơi đều được giải quyết với những ngầm định mà thầy giáo và nhóm đều thống nhất:
  + Mỗi tàu được đánh số khác nhau, bắt đầu từ 1, không phụ thuộc vào kích thước. VD: Tàu đánh số 3 có thể kích thước 2 ô.
  + Yêu cầu nhập kích thước bản đồ, tạo ra 2 file .txt toàn là nước (vùng biển của người chơi), sau đó người chơi có thể đặt tàu bằng cách chỉnh sửa các file .txt này.
  + Quy định số lượt chơi K = N\*N/2, đảm bảo tính hữu hạn, hợp lí, không gây nhàm chán.
  + Đầy đủ các chế độ chơi người với người, người với máy.
* Các yêu cầu nâng cao, mở rộng được tiếp cận:
  + Kích thước bản đồ cho phép tương đối lớn, chưa thể kiểm tra.
  + Kích thước tàu là tùy chọn, do người chơi thỏa thuận, có thể dài hơn 3 ô.
  + Số loại tàu là tùy chọn, do người chơi thỏa thuận.
  + Số điểm đem lại bằng kích thước tàu bị bắn chìm trong lượt.
  + Tăng tính sinh động của trò chơi bằng việc thêm màu sắc và âm thanh.
  + Phát triển chế độ online hai người chơi trên cùng một mạng.

## **2.2. PHÂN CHIA CÔNG VIỆC**

* Team 1 gồm có Nguyễn Trọng Khánh, Bùi Huy Đức sẽ đảm nhiệm các phần: SelectMode(), EmptyMap(), LoadMap(), DrawBoard(), Result().
* Team 2 gồm có Nguyễn Xuân Khải, Lương Minh Kiên sẽ đảm nhiệm các phần: UserInputAttack(), IsNewShip(), SizeOfShip(), CreatMapAI(), GameOverCheck(), Attacking(). Trong đó Khải là người tổng hợp toàn bộ chương trình, đảm bảo chương trình hoạt động tốt.
* Team 3 gồm có Lê Công Nhật Anh, Dương Văn Dũng sẽ đảm nhiệm các phần mở rộng bao gồm chế độ chơi online, phát triển thư viện socketKhai.h, UserInputAttackOnline(), âm thanh, màu sắc.

Kết quả của từng nhóm đều rất tốt, đảm bảo chương trình con cũng như chương trình chính chạy ổn định, chính xác.

# CHƯƠNG 3: THIẾT KẾ VÀ THỰC HIỆN CHƯƠNG TRÌNH

## 3.1. CÁC CHẾ ĐỘ CHƠI

Ta sẽ thiết kế 3 chế độ chơi khác nhau:

* Chơi offline 2 người:

Ở chế độ này, 2 người chơi sẽ tạo sẵn file bản đồ vị trí tàu của riêng mình và lần lượt nhập vào toạ độ tấn công tàu của đối thủ. Hai người sẽ thay phiên nhau chơi đến khi game kết thúc.

* Chơi offline giữa người và máy tính:

Ở chế độ này, người chơi sẽ tạo sẵn file bản đồ vị trí tàu của mình. Từ đó, Máy tính sẽ đọc thông tin số lượng và kích thước tàu rồi tạo file bản đồ riêng. Người chơi và máy tính (bằng thuật toán cài đặt sẵn) lần lượt nhập vào toạ độ tấn công tàu của đối thủ. Cứ như vậy thay phiên nhau chơi đến khi game kết thúc.

* Chơi online 2 người;

Chế độ này hoàn toàn tương tự chế độ chơi thứ nhất, có điều 2 người chơi sẽ chơi trên 2 máy tính khác nhau thông qua kết nối Socket. Trong đó 1 máy sẽ là Server (máy chủ), 1 máy là Client (máy khách).

## 3.2. CÁC THƯ VIỆN, HẰNG, BIẾN DỮ LIỆU

## 3.2.1. Thư viện

#pragma comment(lib, "winmm.lib")

#pragma comment(lib, "ws2\_32.lib")

#include <iostream>

#include <string>

#include <fstream>

#include <stdlib.h> /\* srand, rand \*/

#include <time.h> /\* time \*/

#include <vector>

#include"socketKhai.h"

#include <windows.h>

Trong đó: *winmm.lib* là thư viện phục vụ cho phát âm thanh.

*ws2\_32.lib* là thư viện phục vụ kết nối Client – Server.

*stdlib.h và time.h* phục vụ việc random toạ độ ở chế độ chơi với máy.

*socketKhai.h* thư viện tự tạo chứa các hàm phục vụ gửi, nhận dữ liệu online

## 3.2.2. Hằng

Để tiện cho việc ghi nhớ và lặp trình, ta khai báo một số hằng sau:

#define FOR\_ALL\_POINT\_I\_J for (int i = 0; i < boardSize; i++)\

for (int j = 0; j < boardSize; j++)

//ten file luu ban do cua 2 ng choi

const string mapPlayer1 = "mapPlayer1.txt";

const string mapPlayer2 = "mapPlayer2.txt";

// ki hieu trong ban do

//some symbols for the game

const char isWATER = '0';

const char isHIT = 'X';

const char isMISS = 'M';

//some colors for the game

const char blue[] = "\x1B[34m";

const char normal[] = "\x1B[0m";

const char bblue[] = "\x1B[94m";

const char red[] = "\x1B[31m";

const char yellow[] = "\x1B[93m";

Ta sử dụng ANSI escape code để in ra màu sắc tuỳ chọn. Ví dụ đoạn code sau sẽ in ra chữ Xin chao màu xanh:

cout << "\x1B[34m" << "Xin chao" << normal << endl;

## 3.2.3. Biến dữ liệu

Ta chỉ liệt kê 2 biến tự định nghĩa:

* Lớp *CORD* gồm 2 biến *x,y* để lưu toạ độ 1 điểm trên bản đồ

class CORD {

public:

int x;

int y;

CORD(int x, int y) {

this->x = x;

this->y = y;

}

};

* Cấu trúc *PLAYER* lưu thông tin người chơi

struct PLAYER {

string map; //ten file ban do

char\*\* grid = NULL; //mang ban do

int boom = 0; //luu so ten lua= so luot choi

int score = 0; //diem

};

## 3.3. CÁC HÀM SỬ DỤNG

* Chọn chế độ chơi

int SelectMode() {

cout << endl << "\n\_\_\_\_\_\_\_" << bblue << "\*\*\*\*" << red << " B A T T L E S H I P S " << bblue << "\*\*\*\*" << normal << "\_\_\_\_\_\_\_\_\_" << endl;

cout << endl << "\n\n GAME MODE:" << endl;

cout << "-------------------------------------";

cout << "\n| 1. HUMAN VS HUMAN OFFLINE MATCH |" << endl;

cout << "| 2. HUMAN VS AI MATCH |" << endl;

cout << "| 3. HUMAN VS HUMAN ONLINE MATCH |" << endl;

cout << "-------------------------------------" << endl;

int ch = 0;

//lap lai cho den khi chon dung 1 trong 3

while (ch != 1 && ch != 2 && ch != 3) {

cout << endl << "To start, please select one game mode... " << endl;

cin >> ch;

}

//tra ve gia tri 1-3

return ch;

}

* Tạo file bản đồ trống:

Nhiệm vụ vủa hàm là tạo sẵn file bản đồ trống kích thước N\*N cho người chơi đặt tàu, hàm được gọi khi chuẩn bị chơi.

Code:

void EmptyMap(PLAYER player) {

ofstream file(player.map);

file << boardSize << endl;

for (int i = 0; i < boardSize; i++) {

for (int j = 0; j < boardSize; j++) {

file << '0' << " ";

}

if (i<boardSize-1) file << endl;

}

file.close();

* Load file bản đồ

bool LoadMap(PLAYER\* player) {

ifstream fileInput(player->map);

if (fileInput.fail()) return false;

// doc kich thuoc

fileInput >> boardSize; //Load the size of map(board)

if (boardSize <= 0) return false;

//tao bo nho dong

//Creat dynamic memory

player->grid = new char\* [boardSize];

for (int i = 0; i < boardSize; i++) {

(player->grid)[i] = new char[boardSize];

}

//doc tung diem tren ban do

//Read each point in map of player

while (!fileInput.eof()) {

FOR\_ALL\_POINT\_I\_J{

fileInput >> player->grid[i][j];

}

}

fileInput.close();

return true;

}

* Kiểm tra điểm đang xét có thuộc 1 tàu mới chưa được duyệt hay không:

bool IsNewShip(vector<char> shipName, char thisPos) {

for (int k = 0; k < shipName.size(); k++)

if ((thisPos == shipName.at(k))) return false;

if (thisPos != isWATER) return true;

return false;

}

* A close up of text on a white background

  Description automatically generatedIn ra bản đồ:

Nhiệm vụ của hàm là in ra bản đồ của đối thủ và lịch sử bắn trúng, trượt của người chơi nhưng không cho biết vị trí các tàu chưa trúng đạn. Hàm này được gọi mỗi khi người chơi nhập toạ độ bắn

Sơ đồ giải thuật:

Code:

void DrawBoard(int thisPlayer, PLAYER player, vector<char> shipName)

{

//Draws the board for a player (thisPlayer)

cout << "\nPLAYER " << thisPlayer << "'s GAME BOARD\n";

cout << "----------------------\n";

//Loop through top row (board\_width) and number columns

cout << " ";

for (int w = 0; w < boardSize; ++w) {

if (w < 10)

//Numbers only 1 character long, add two spaces after

cout << w << " ";

else if (w >= 10)

//Numbers 2 characters long, add only 1 space after

cout << w << " ";

}

cout << "\n";

//Loop through each grid point and display to console

for (int h = 0; h < boardSize; ++h) {

for (int w = 0; w < boardSize; ++w) {

//If this is the FIRST (left) grid point, number the grid first

if (w == 0) cout << h << " ";

//If h was 1 character long, add an extra space to keep numbers lined up

if (w < 10 && w == 0) cout << " ";

//Don't show ships, BUT show damage if it's hit

bool isShip = false;

//kiem tra diem dang xet co la tau hay khong

for (int i = 0; i < shipName.size(); i++)

if (player.grid[h][w] == shipName.at(i)) {

isShip = true;

break;

}

if (!isShip)

{

if (player.grid[h][w] == isHIT) cout << red << player.grid[h][w] << normal << " ";

else if (player.grid[h][w] == isMISS) cout << yellow << player.grid[h][w] << normal << " ";

else cout << bblue << player.grid[h][w] << normal << " ";

}

//neu la tau in ra nuoc

else cout << bblue << isWATER << normal << " ";

//If we have reached the border.. line feed

if (w == boardSize - 1) cout << "\n";

}

}

cout << "----------------------\n";

}

* Nhập toạ độ tấn công trong chế độ offline

A close up of a piece of paper

Description automatically generatedNhiệm vụ của hàm là lấy toạ độ tấn công từ người chơi hoặc tạo toạ độ tấn công nếu là lượt chơi của máy tính.

Hàm được gọi trong mỗi lượt chơi sau khi gọi hàm *DrawBoard().*

Sơ đồ giải thuật:

Code:

bool UserInputAttack(int& x, int& y, int theplayer, int gameMode, PLAYER enemyPlayer)

{

bool goodInput = false;

//danh voi may

if (gameMode == 2 && theplayer == 2) {

cout << "\nAI IS ATTACKING ..";

//Your mission is to creat x,y for AI

Sleep(1000);

cout << "..";

Sleep(1000);

cout << "..";

Sleep(1000);

{

srand(time(NULL));

vector<CORD> cord; //save all empty point to a vector to select from

//thuat toan chon diem ban cua may

//dau tien chon trong so cac o den

//first save all black points

FOR\_ALL\_POINT\_I\_J

if (enemyPlayer.grid[i][j] != isHIT && enemyPlayer.grid[i][j] != isMISS)

if (((i % 2 == 1) && (j % 2 == 1)) || ((i % 2 == 0) && (j % 2 == 0)))

cord.push\_back(CORD(i, j));

//neu het o den thi chon cac o con lai mot cach ngau nhien

//if there are no black points, choose normally

if (cord.size() == 0) {

FOR\_ALL\_POINT\_I\_J

if (enemyPlayer.grid[i][j] != isHIT && enemyPlayer.grid[i][j] != isMISS)

cord.push\_back(CORD(i, j));

}

//chon ngau nhien toa do x,y

int hunt = rand() % cord.size(); //random a point in vector

x = cord.at(hunt).x;

y = cord.at(hunt).y;

}

goodInput = true;

return goodInput;

}

//phan cho nguoi choi nhap

cout << "\nPLAYER " << theplayer << ", ENTER COORDINATES TO ATTACK: ";

cin >> x >> y; //the first one is horizontal

//kiem tre nguoi nhap diem hop le hay khong

if (x < 0 || x >= boardSize) return goodInput;

if (y < 0 || y >= boardSize) return goodInput;

goodInput = true;

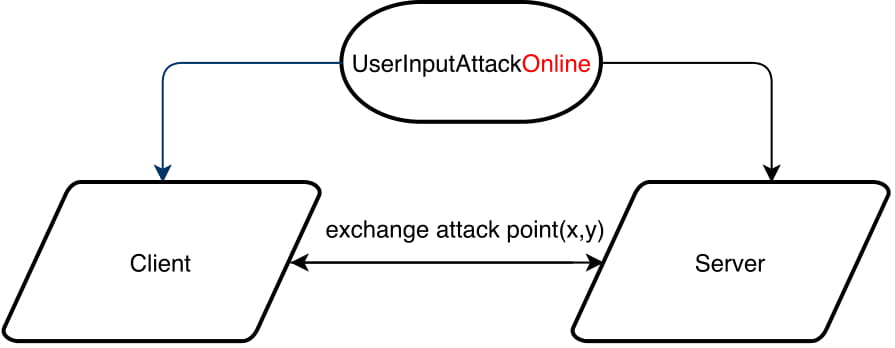
return goodInput;

}

* Nhập toạ độ tấn công trong chế độ online

Nhiệm vụ là gửi, nhận toạ độ tấn công giữa 2 máy tính trong quá trình chơi. Tương tự như hàm *UserInputAttack(),* hàm này cũng được gọi sau mỗi lượt chơi sau hàm *DrawBoard().*

Sơ đồ giải thuật:



Code:

bool UserInputAttackOnline(int& x, int& y, int thePlayer, SOCKET server, SOCKET client)

{

bool goodInput = false;

SOCKET target;

int turn;

if (isServer == 1) { target = client; turn = 1; }

else { target = server; turn = 2; }

//if it is your turn

if (thePlayer == turn) {

cout << "\nPLAYER " << thePlayer << ", ENTER COORDINATES TO ATTACK: ";

cin >> x >> y; //the first one is horizontal

if (x < 0 || x >= boardSize) return goodInput;

if (y < 0 || y >= boardSize) return goodInput;

SendCord(target, x, y);

goodInput = true;

}

//if it is your enemy turn

else {

cout << "\nPLAYER " << thePlayer << " IS ENTERING COORDINATES TO ATTACK...\n-----PLEASE WAIT-----";

Sleep(1000);

ReceiveCord(target, x, y);

goodInput = true;

}

return goodInput;

}

* Kiểm tra game kết thúc:

Nhiệm vụ của hàm là kiểm tra bản đồ, điểm số, lượt chơi của mỗi người sau mỗi lượt tấn công và đưa ra kết quả. Từ đó cho biết trò chơi đã kết thúc chưa, nếu rồi thì người chơi nào thắng.

Sơ đồ giải thuật:

A close up of a map

Description automatically generated

Code:

int GameOverCheck(PLAYER thisPlayer, int thisIndex, PLAYER enemyPlayer, int enemyIndex, vector<char> shipName)

{

int winner = thisIndex;

//Loop through enemy board

for (int w = 0; w < boardSize; ++w) {

for (int h = 0; h < boardSize; ++h) {

//If any ships remain, game is NOT over

for (int k = 0; k < shipName.size(); k++) if (shipName.at(k) == enemyPlayer.grid[h][w]) winner = 0;

}

}

//kiem tra so luot choi

if ((enemyPlayer.boom == 0) && (thisPlayer.boom == 0))

if (thisPlayer.score > enemyPlayer.score) winner = thisIndex;

else if (thisPlayer.score < enemyPlayer.score) winner = enemyIndex;

else {

winner = 3;

}

//If we get here, thisPlayer won, game over!

return winner;

}

* In ra kết quả của trò chơi:

Sau khi trò chơi kết thúc, hàm này sẽ có nhiệm vụ in ra kết quả dựa vào giá trị trả về của hàm *GameOverCheck()*, tương ứng 3 trường hợp :Thắng, Thua, Hoà.

Ở đây ta sử dụng hàm *Play Sound* trong thư viện winmm.lib để phát 1 đoạn nhạc kết thúc

PlaySound(TEXT(char\* filename), NULL, SND\_FILENAME | SND\_ASYNC);

Code:

void Result(int aWin, PLAYER\* player, int gameMode) {

system("cls");

cout << yellow;

if (aWin == 3) {

cout << "\n\n WE HAVE A TIE GAME!!! \n\n\n\n";

}

if (gameMode == 1 && aWin != 3) {

PlaySound(TEXT("win.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

cout << "\n\n CONGRATULATIONS!!! PLAYER " << aWin << " HAS WON THE GAME!\n\n\n\n";

}

if (gameMode == 2) {

if (2 == aWin) {

PlaySound(TEXT("lose.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

cout << "\n\n YOU HAVE LOST TO AI!!! \n\n\n\n";

}

if (1 == aWin) {

PlaySound(TEXT("win.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

cout << "\n\n CONGRATULATIONS!!! YOU HAVE WON AI!!! \n\n\n\n";

}

}

if (gameMode == 3) {

if (aWin == notServer) {

PlaySound(TEXT("lose.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

cout << "\n\n YOU LOSE! ENEMY PLAYER HAS WON THE GAME!\n\n\n\n";

}

if (aWin == isServer) {

PlaySound(TEXT("win.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

cout << "\n\n CONGRATULATIONS!!! YOU HAS WON THE GAME!\n\n\n\n";

}

}

cout << normal;

cout << "\nHope you enjoy this game. Thanks for your time.\n\n";

system("pause");

}

* Tính toán kích thước tàu:

int SizeOfShip(PLAYER player1, char shipName) {

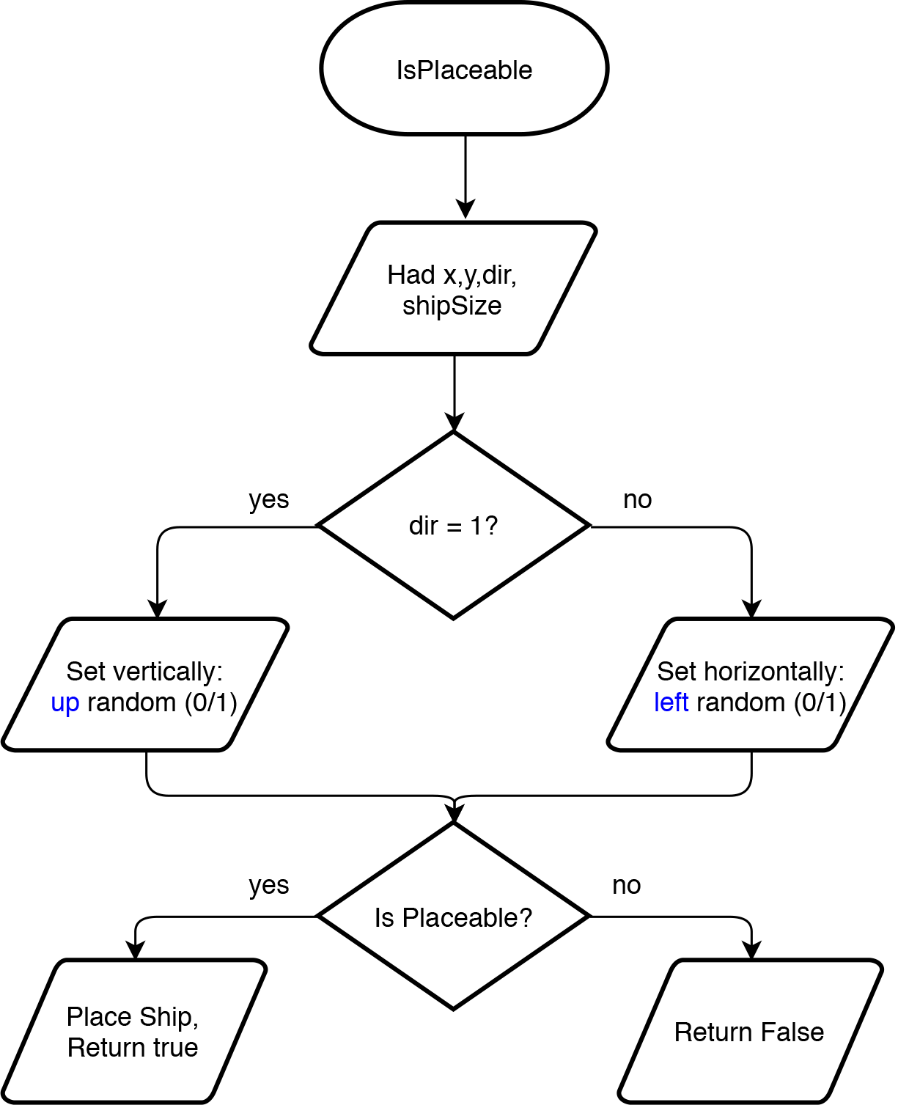
int count = 0;

FOR\_ALL\_POINT\_I\_J if (player1.grid[i][j] == shipName) count++;

return count;

}

* Kiểm tra vị trí đang xét có đặt được tàu không (dùng khi máy tạo bản đồ):

Sơ đồ giải thuật:

Code:

bool IsPlaceable(int x, int y, int dir, PLAYER& player2, char shipName, int shipSize) {

bool logic = true;

srand(time(NULL));

if (dir == 0)

{

int left = rand() % 2;

if (left == 1) {

for (int i = y - shipSize + 1; i < y + 1; i++)

if (0 <= i && i < boardSize)

if (player2.grid[x][i] != isWATER) {

logic = false;

break;

}

else {

logic = false;

break;

}

if (logic == true) for (int i = y - shipSize + 1; i < y + 1; i++) player2.grid[x][i] = shipName;

}

else {

for (int i = y; i < y + shipSize; i++)

if (0 <= i && i < boardSize)

if (player2.grid[x][i] != isWATER) {

logic = false;

break;

}

else {

logic = false;

break;

}

if (logic == true) for (int i = y; i < y + shipSize; i++) player2.grid[x][i] = shipName;

}

}

if (dir == 1)

{

int up = rand() % 2;

if (up == 0) {

for (int i = x - shipSize + 1; i < x + 1; i++)

if (0 <= i && i < boardSize)

if (player2.grid[i][y] != isWATER) {

logic = false;

break;

}

else;

else {

logic = false;

break;

}

if (logic == true) for (int i = x - shipSize + 1; i < x + 1; i++) player2.grid[i][y] = shipName;

}

else {

for (int i = x; i < x + shipSize; i++)

if (0 <= i && i < boardSize)

if (player2.grid[i][y] != isWATER) {

logic = false;

break;

}

else;

else {

logic = false;

break;

}

if (logic == true) for (int i = x; i < x + shipSize; i++) player2.grid[i][y] = shipName;

}

}

return logic;

}

* Hàm tạo bản đồ cho máy tính:

A close up of a map

Description automatically generatedNhiệm vụ của hàm là dựa vào thông tin số lượng, kích thước tàu lấy từ bản đồ của người chơi để tạo nên bản đồ mới cho máy.

Sơ đồ thuật toán:

Code:

void CreatMapAI(PLAYER player1, PLAYER& player2, vector<char> shipName) {

//Your mission is to creat a mapPlayer2.txt which has AI's map, depending on mapPlayer1.txt

vector<int> shipSize; //take all the sizes of ships to a equivalent vector like shipName

//luu vecto chua kich thuoc cua cac tau

for (int thisShip = 0; thisShip < shipName.size(); thisShip++) shipSize.push\_back(SizeOfShip(player1, shipName.at(thisShip)));

srand(time(NULL));

int x, y, dir;

for (int thisShip = 0; thisShip < shipName.size(); thisShip++) {

do {

x = rand() % boardSize;

//the first set point of a ship

y = rand() % boardSize;

dir = rand() % 2;

//0 is horizontal, 1 is vertical

} while (!IsPlaceable(x, y, dir, player2, shipName.at(thisShip), shipSize.at(thisShip)));

}

}

* Cài đặt ban đầu cho chế độ online:

Gồm 2 hàm *ServerConfig()* và *ClientConfig().*

Nhiệm vụ của hàm là khởi tạo socket, đợi kết nối của máy khách (đối với Server) hoặc kết nối đến máy chủ (đối với Client). Sau khi kết nối thành công, hai hàm sẽ thực hiện gửi file bản đồ của mình và nhận file bản đồ của đối phương.

***Giới thiệu Socket và cơ chế giao tiếp Client – Server****:*

* + *Giới thiệu về Socket:*

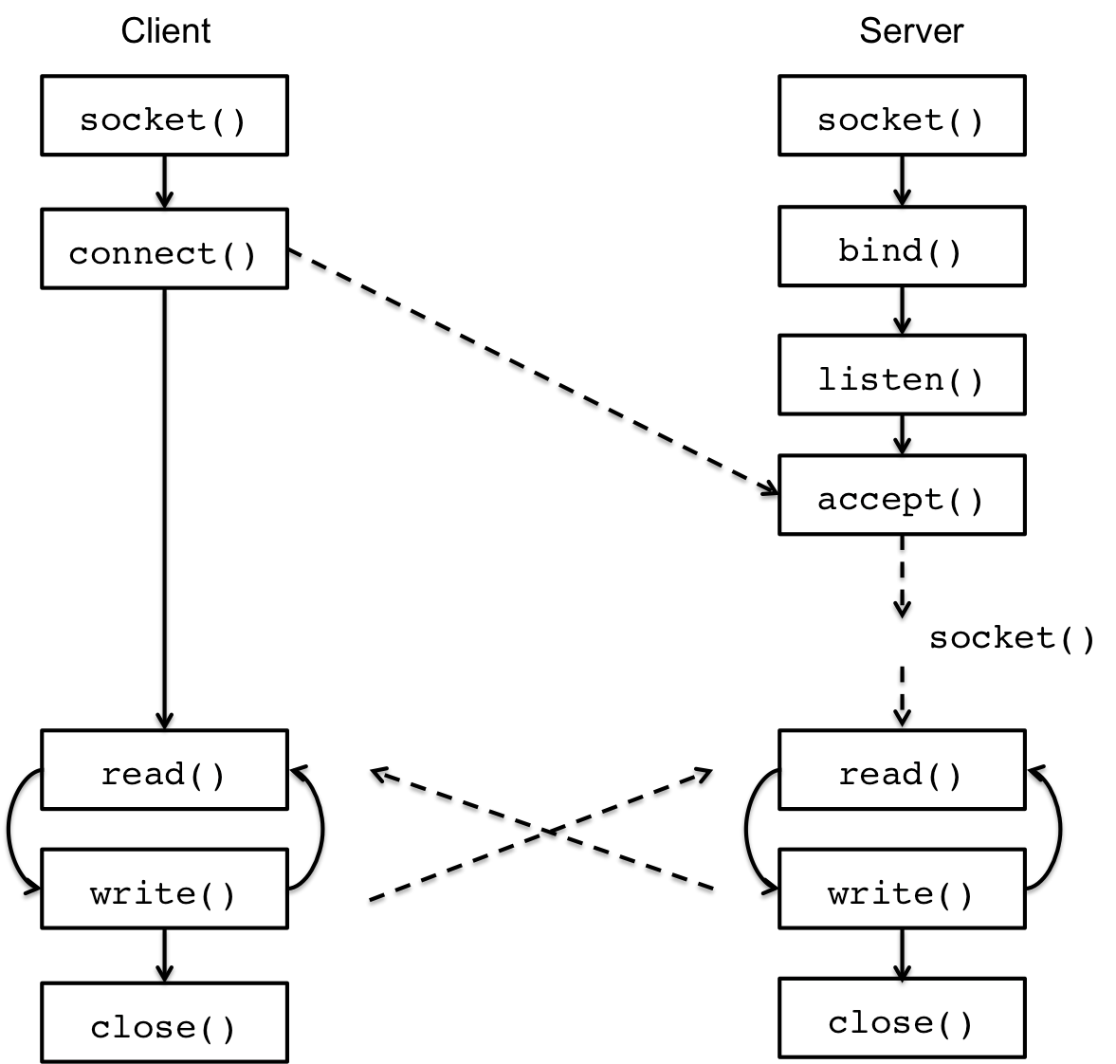
Socket là một giao diện lập trình (API – Application Program Interface) ứng dụng mạng thông qua giao diện này có thể lập trình điều khiển việc truyền thông giữa 2 máy sử dụng các giao thức mức thấp như TCP,UDP… , Có thể tưởng tượng nó như một thiết bị truyền thông 2 chiều tương tự như tệp tin, chúng ta gửi/nhận dữ liệu giữa 2 máy, tương tự như việc đọc ghi trên tệp tin.

Có 2 loại socket được sử dụng rộng rãi là: *Stream sockets* và *Datagram sockets.*

- *Stream sockets:* Dựa trên giao thức TCP (Tranmission Control Protocol), là giao thức hướng luồng (stream oriented). Việc truyền dữ liệu chỉ thực hiện giữa 2 tiến trình **đã thiết lập kết nối**. Giao thức này đảm bảo dữ liệu được truyền đến nơi nhận một cách đáng tin cậy, đúng thứ tự nhờ vào cơ chế quản lý luồng lưu thông trên mạng và cơ chế chống tắc nghẽn.

- *Datagram sockets*: Dựa trên giao thức UDP (User Datagram Protocol), là giao thức hướng thông điệp (message oriented). Việc truyền dữ liệu **không yêu cầu có sự thiết lập kết nối giữa tiến quá trình**. Ngược lại với giao thức TCP thì dữ liệu được truyền theo giao thức UDP không được tin cậy, có thế không đúng trình tự và lặp lại. Tuy nhiên vì nó không yêu cầu thiết lập kết nối không phải có những cơ chế phức tạp nên tốc độ nhanh…ứng dụng cho các ứng dụng truyền dữ liệu nhanh như chat, game…..

* + *Mô hình khách chủ TCP:*



* + - Phía Server:

*Các bước thực hiện:*

* Tạo socket với hàm socket (int family, int type, int protocol) các tham số trong đó theo thứ tự là họ giao thức, kiểu socket, kiểu giao thức.
* Gán địa chỉ cho socket bind (int sockfd, const struct sockaddr \*sockaddr, socklen\_t addrlen) các tham số lần lượt là mô tả socket vừa tạo, con trỏ chỏ đến địa chỉ socket, độ lớn địa chỉ
* Chỉ định socket lắng nghe kết nối listen (int sockfd, int backlog) trong đó sockfd là mô tả socket vừa tạo, backlog là số lượng tối đa các kết nối đang chờ
* Chờ/chấp nhận kết nối accept (int sockfd, struct sockaddr \*cliaddr, socklen\_t \*addrlen) lần lượt có các tham số là mô tả socket vừa tạo, con trỏ tới cấu trúc địa chỉ socket của tiến trình kết nối đến, độ lớn cấu trúc địa chỉ
* Gửi, nhận dữ liệu

Code:

bool ServerConfig(SOCKET& client, SOCKET& server) {

WSADATA WSAData;

SOCKADDR\_IN serverAddr, clientAddr;

bool run = false;

WSAStartup(MAKEWORD(2, 0), &WSAData);

server = socket(AF\_INET, SOCK\_STREAM, 0);

serverAddr.sin\_addr.s\_addr = INADDR\_ANY;

serverAddr.sin\_family = AF\_INET;

serverAddr.sin\_port = htons(5555);

bind(server, (SOCKADDR\*)&serverAddr, sizeof(serverAddr));

listen(server, 0);

system("cls");

system("C:\\Windows\\System32\\ipconfig"); //print out ipconfig console, to see all IP

cout << red<<"\nTELL YOUR RIVAL YOUR IPV4 ADDRESS, WHICH APPEARS IN THE FIRST SCREEN" << normal<< endl;

cout << "\nListening for incoming connections..." << endl;

int clientAddrSize = sizeof(clientAddr);

if ((client = accept(server, (SOCKADDR\*)&clientAddr, &clientAddrSize)) != INVALID\_SOCKET)

{

cout << "\nClient connected!" << endl;

run = true;

}

else {

cout << "\nCan't connect to client" << endl;

run = false;

}

cout << "\nPLEASE CREATE YOUR OWN FLEE IN TXT FILE !!! " << endl;

system("pause");

ReceiveFile(client, "mapPlayer2.txt");

Sleep(500);

SendFile(client, "mapPlayer1.txt");

Sleep(500);

return run;

}

* + - Phía Client

*Các bước thực hiện:*

* Tạo socket với hàm socket (int family, int type, int protocol) các tham số trong đó theo thứ tự là họ giao thức, kiểu socket, kiểu giao thức.
* Connect tới địa chỉ server với hàm connect(sock, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr)
* Gửi, nhận dữ liệu

Code:

bool ClientCofig(SOCKET& client, SOCKET& server) {

WSADATA WSAData;

SOCKADDR\_IN addr;

string serverIP;

WSAStartup(MAKEWORD(2, 0), &WSAData);

server = socket(AF\_INET, SOCK\_STREAM, 0);

system("cls");

serverIP.assign(GetIP()); //get IP of server, server must TELL client

addr.sin\_addr.s\_addr = inet\_addr(serverIP.c\_str()); // replace the ip with your futur server ip address. If server AND client are running on the same computer, you can use the local ip 127.0.0.1

addr.sin\_family = AF\_INET;

addr.sin\_port = htons(5555);

connect(server, (SOCKADDR\*)&addr, sizeof(addr));

cout << "\nConnected to server!" << endl;

cout << "\nPLEASE CREATE YOUR OWN FLEE IN TXT FILE !!! " << endl;

system("pause");

SendFile(server, "mapPlayer2.txt");

Sleep(500);

ReceiveFile(server, "mapPlayer1.txt");

Sleep(500);

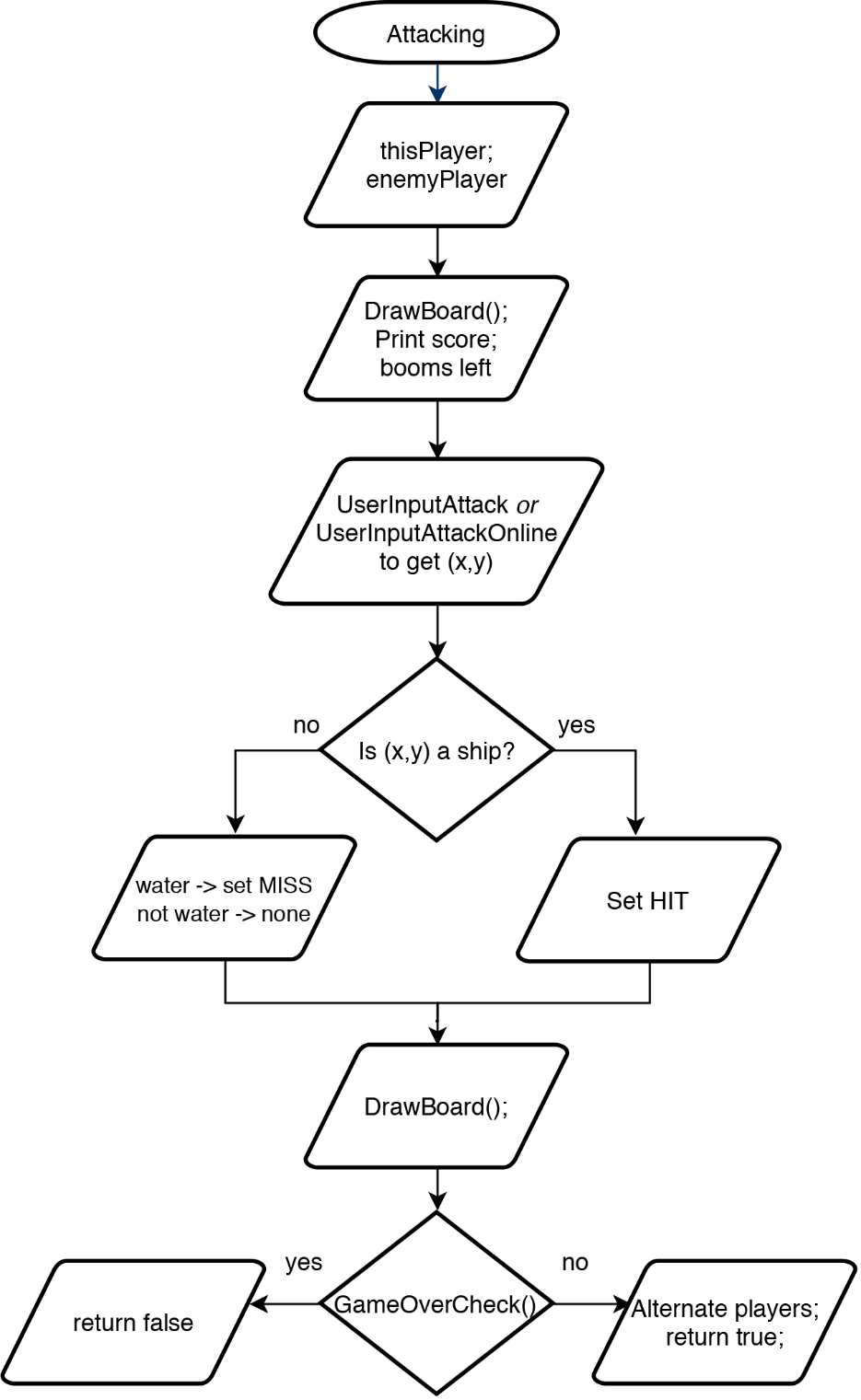
return true;

}

* Hàm thực hiện lượt chơi:

Đây là hàm quan trọng nhất, thực hiện mọi thao tác trong mỗi lượt chơi của mỗi người.

Sơ đồ giải thuật:



Code:

bool Attacking(int& thisPlayer, int& aWin, vector<char>& shipName, PLAYER\* player, int gameMode) {

//Because we are ATTACKING now, the opposite players board is the display board

int enemyPlayer;

//xac dinh luot choi

if (thisPlayer == 1) enemyPlayer = 2;

if (thisPlayer == 2) enemyPlayer = 1;

system("cls");

//ve map

DrawBoard(enemyPlayer, player[enemyPlayer], shipName);

//in ra diem va so luot

cout << endl << "PLAYER " << thisPlayer << "'S SCORE: " << player[thisPlayer].score << endl;

cout << "PLAYER " << thisPlayer << "'S BOOMS LEFT: " << player[thisPlayer].boom << endl;

//Get attack coords from this player

//nhan toa do tan cong

bool goodInput = false;

int x = 0, y = 0;

while (goodInput == false) {

if (gameMode == 3) goodInput = UserInputAttackOnline(x, y, thisPlayer, server, client);

else goodInput = UserInputAttack(x, y, thisPlayer, gameMode, player[enemyPlayer]);

}

//giam di 1 luot

player[thisPlayer].boom--;

//kiem tra toa do ban

//Check board; if a ship is there, set as HIT.. otherwise MISS

for (int thisShip = 0; thisShip < shipName.size(); thisShip++)

if (player[enemyPlayer].grid[x][y] == shipName.at(thisShip))

{

//nice shot

PlaySound(TEXT("Explosion+2.wav"), NULL, SND\_FILENAME | SND\_ASYNC);

//Sleep(3500);

FOR\_ALL\_POINT\_I\_J

if (player[enemyPlayer].grid[i][j] == shipName.at(thisShip)) {

player[enemyPlayer].grid[i][j] = isHIT;

player[thisPlayer].score++;

} break;

}

if (player[enemyPlayer].grid[x][y] == isWATER) player[enemyPlayer].grid[x][y] = isMISS;

system("cls");

//ve ban do sau khi ban

DrawBoard(enemyPlayer, player[enemyPlayer], shipName);

cout << endl;

system("pause");

//Check to see if the game is over

//If 0 is returned, nobody has won yet

aWin = GameOverCheck(player[thisPlayer], thisPlayer, player[enemyPlayer], enemyPlayer, shipName);

if (aWin != 0) return false;

//Alternate between each player as we loop back around

thisPlayer = (thisPlayer == 1) ? 2 : 1;

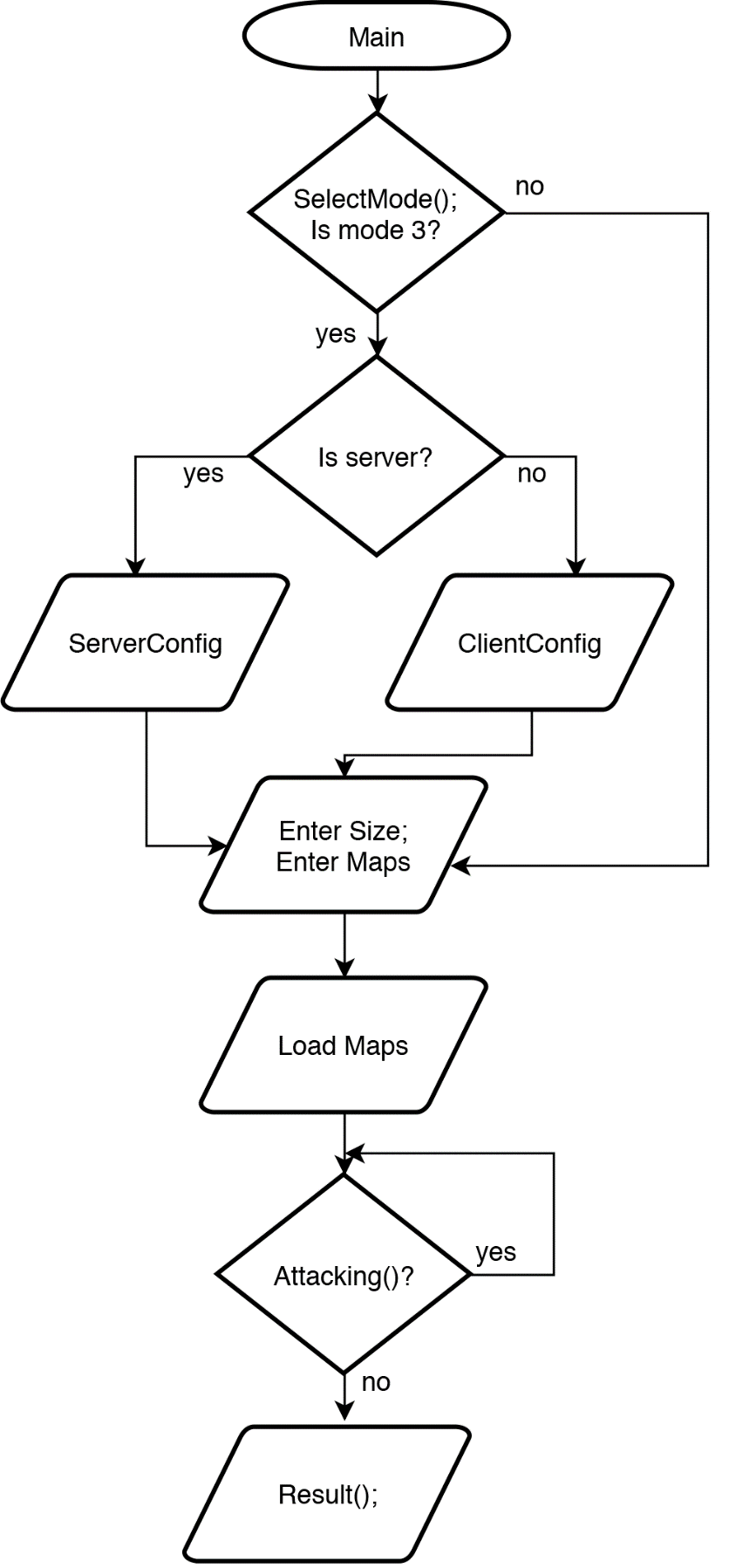
return true;

}

3.4 Hàm *main()*

Hàm *main()* có thực hiện các thao tác khởi tạo bạn đầu và liên kết các hàm, thủ tục ở trên thành một chương trình thống nhất.

Sơ đồ giải thuật:



Code:

int main()

{

PLAYER\* player = new PLAYER[3]; //Use player 1 & 2, 0 is ignored

player[1].map = mapPlayer1;

player[2].map = mapPlayer2;

int thisPlayer = 1;

int aWin = 0;

int gameMode = SelectMode();

//che do online

if (gameMode == 3) {

//chon lam server, hay client

cout << "\n<If you are player 1, you will be server. Otherwise, if you are player 2, you will be client !!>" << endl

<< "\nWhich player do you want to be (1/2)?"<<endl;

cin >> isServer;

notServer = (isServer == 1) ? 2 : 1;

cout << "\nPLEASE ENTER SIZE OF MAP: ";

cin >> boardSize;

EmptyMap(player[1]);

EmptyMap(player[2]);

if (isServer == 1) ServerConfig(client, server); else ClientCofig(client, server);

if (!LoadMap(&player[isServer])) {

cout << "\nFailed to load player " << isServer << "'s map!" << endl;

return -1;

}

vector<char> shipName; //vector saves types of ship

shipName.push\_back('1');

FOR\_ALL\_POINT\_I\_J{

if (IsNewShip(shipName, player[isServer].grid[i][j])) shipName.push\_back(player[isServer].grid[i][j]);

}

player[isServer].boom = (player[notServer].boom = boardSize \* boardSize / 2); //config the number of turns = boom

if (!LoadMap(&player[notServer])) {

cout << "\nFailed to load player " << notServer << "'s map!" << endl;

return -1;

}

cout << "\nPLAYERS' MAPS LOADED, READY TO FIGHT!!!" << endl;

system("pause");

//Ready to play the game

do {} while (Attacking(thisPlayer, aWin, shipName, player, gameMode));

}

// choi che do offline

else {

cout << "\nPLEASE ENTER SIZE OF MAP: ";

cin >> boardSize;

EmptyMap(player[1]);

EmptyMap(player[2]);

cout << "\nPLEASE SECRETLY CREATE YOUR OWN FLEE IN TXT FILE !!! " << endl;

system("pause");

if (!LoadMap(&player[1])) {

cout << "\nFailed to load player 1's map!" << endl;

return -1;

}

vector<char> shipName; //vector saves types of ship

shipName.push\_back('1');

FOR\_ALL\_POINT\_I\_J{

if (IsNewShip(shipName, player[1].grid[i][j])) shipName.push\_back(player[1].grid[i][j]);

}

player[1].boom = (player[2].boom = boardSize \* boardSize / 2); //config the number of turns = boom

//neu choi voi may thi tao map trong cho may

if (gameMode == 2) {

player[2].grid = new char\* [boardSize];

for (int i = 0; i < boardSize; i++) {

(player[2].grid)[i] = new char[boardSize];

}

FOR\_ALL\_POINT\_I\_J player[2].grid[i][j] = isWATER;

cout << "\nAI IS CREATING ITS FLEE ON THE MAP. PLEASE WAIT A MINUTE...\n";

//cho AI tao map

CreatMapAI(player[1], player[2], shipName);

}

else if (!LoadMap(&player[2])) {

cout << "\nFailed to load player 2's map!" << endl;

return -1;

}

cout << "\nPLAYERS' MAPS LOADED, READY TO FIGHT!!!" << endl;

system("pause");

//Ready to play the game

do {} while (Attacking(thisPlayer, aWin, shipName, player, gameMode));

}

//in ra ket qua

Result(aWin, player, gameMode);

delete[](player[1].grid);

delete[](player[2].grid);

return 0;

# CHƯƠNG 4: CÀI ĐẶT CHƯƠNG TRÌNH

## **4.1. ƯU ĐIỂM**

* Lập trình đơn giản, dễ nắm bắt cho lập trình viên.
* Chỉ sử dụng các thư viện có sẵn, và chỉ thêm một thư viện tự tạo để thuận tiện cho việc lập trình.
* Giao diện chương trình đơn giản, dễ sử dụng, dễ thao tác tạo hứng thú cho người sử dụng.
* Chương trình sinh động, thích hợp như một game giải trí cũng như cạnh tranh giữa 2 người chơi.

## **4.2. NHƯỢC ĐIỂM**

* Chưa phát huy được sức mạnh Hướng Đối Tượng của C++.
* Chưa phát triển đồ họa, âm thanh chuyên nghiệp.
* Thuật toán của máy chơi chưa tối ưu.

## **4.3. HƯỚNG PHÁT TRIỂN CỦA NHÓM**

* Phát triển theo hướng lập trình Hướng Đối Tượng.
* Áp dụng trí thông minh nhân tạo cho máy.
* Thiết kế đồ họa và âm thanh chuyên nghiệp.

# TÀI LIỆU THAM KHẢO

1. Microsoft C/C++ language and standard libraries reference.

<https://docs.microsoft.com/en-us/cpp/cpp/c-cpp-language-and-standard-libraries?view=vs-2019>

1. Diễn đàn hỏi đáp lập trình.

<https://stackoverflow.com/>

1. Và một số trang web dạy lập trình khác.