**Battle Ship**

Project 1

Lei Wan

5-1-2016

**Content**

1. **Intruduce**
   1. **Rules**
2. **Development**
   1. **Difficult points**
   2. **AI**
   3. **Knowledge points**
3. **Pseudo Code**
4. **Flowchart**
5. **Code**
6. **Introduction**
   1. Rules

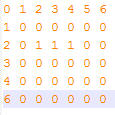
Battle ship is a classic board game. Player and Computer both have five ships, and each ship’s length is different (from one to five). Player and computer need to put the five ships on a board before the game start. After that, Player and computer take turns to choose a location on the board, if there is a ship on that location, who made the correct choice could do more choice

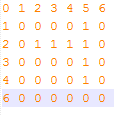
1. **Development**
   1. Difficult points.

1. Pure virtual function. I set a class Bomb as abstract class, and make the class BombS and BombN inherit from it.

2.When the ships on the board appear cross, like “T”. General AI need make some change.

2.2

 in this situation, if the first grid we chose is (2,4),we will check (2,3) and (2,5).

in this situation , if we find (2,2),(2,3),(2,4),(2,5),then we find (3,5), generally,We will check (2,5) and (4,5). When we find (2,5), has been found, we need jump this one, and check (1,5). I realize this by recusion.

2.3 knowledge points

1.binary read and write

2.dymatic location

3.class array

4.abstract class (pure virtual method)

5.cstring strcpy(char\*,string.c\_str())

6.friend class

**3. Pseudo Code**

Set rand number;

Do{

Display board;

If(no ship been found){

Do{

Rand the (x,y) point;

(x,y) is suitable;

}while(suitable jump out)

If(no ship on this point)

Set flag( no ship);

else{

set flag(be hit);

recorde the (x,y) to 2 group of viarable(x,y),(x1,y1);

set flag(hit another time);

}

else {

if(one grid of ship be hit){

do{

rand a number(0-3), the ship bow direction;

if(toward south)

if(x<=0)

set flag(invalid dirction);

else{

if((x-1,y)point no ship){

set flag(no ship);

set flag(hit another time);

}

else if(x-1,y)is 1 or 3)

set flag(invalid pint);

else{

set flag(hit);

x1=x-1;

y1=y;

set flag(hit again);

}

}

else if(towards west){

if(touch the bound)

set flag(invalid direction);

else {

if(no ship there ){

set flag(no ship);

set flag(not hit again);

}else if(ther flage no ship there or flag be hit ther){

Set flag(hit agin);

}else{

Set flag(be hit);

X1=x; y1=y+1;

Set flag(hit again);

}

}

Else if(towards noth){

If(bound)

Set flag(invalid direct);

Else{

If (no ship there){

Set flag(no ship there);

Set flag(no hit again);

}else if(flag(no ship there or flag(be hit there){

Set flag(hit again);

} else{

Set flag(be hit there)

X1=x+1;

Y1=y;

Set flag(hit again);

}

}

}else{

If(bound)

Set flag(invalid direction);

Else{

If(no ship there){

Set flag(no ship ther);

Set flag(not hit again);

}else if(flag(no ship ther) or flag(be

Hit ther))

Set flag(hit again);

Else {

Set flag(be hit ther);

X1=x; y1=y-1;

Set flag(hit again);

}

}

} while(invalid random number)

}else{

If(no change check direction)

Call shipsRecur()

Else if(change check dirction and the first change){

Swap(x,x1);

Swap(y,y1);

Call shipsRecur();

}else{

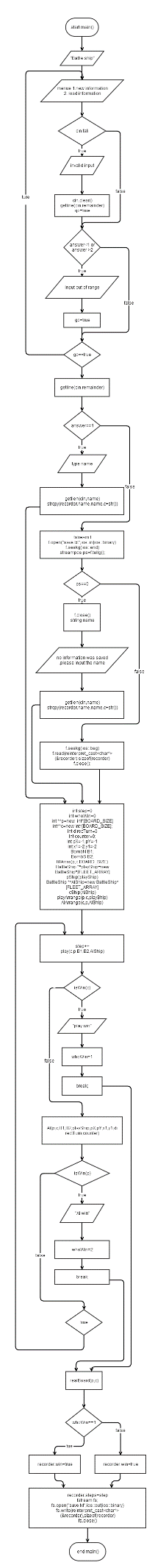
Reset;

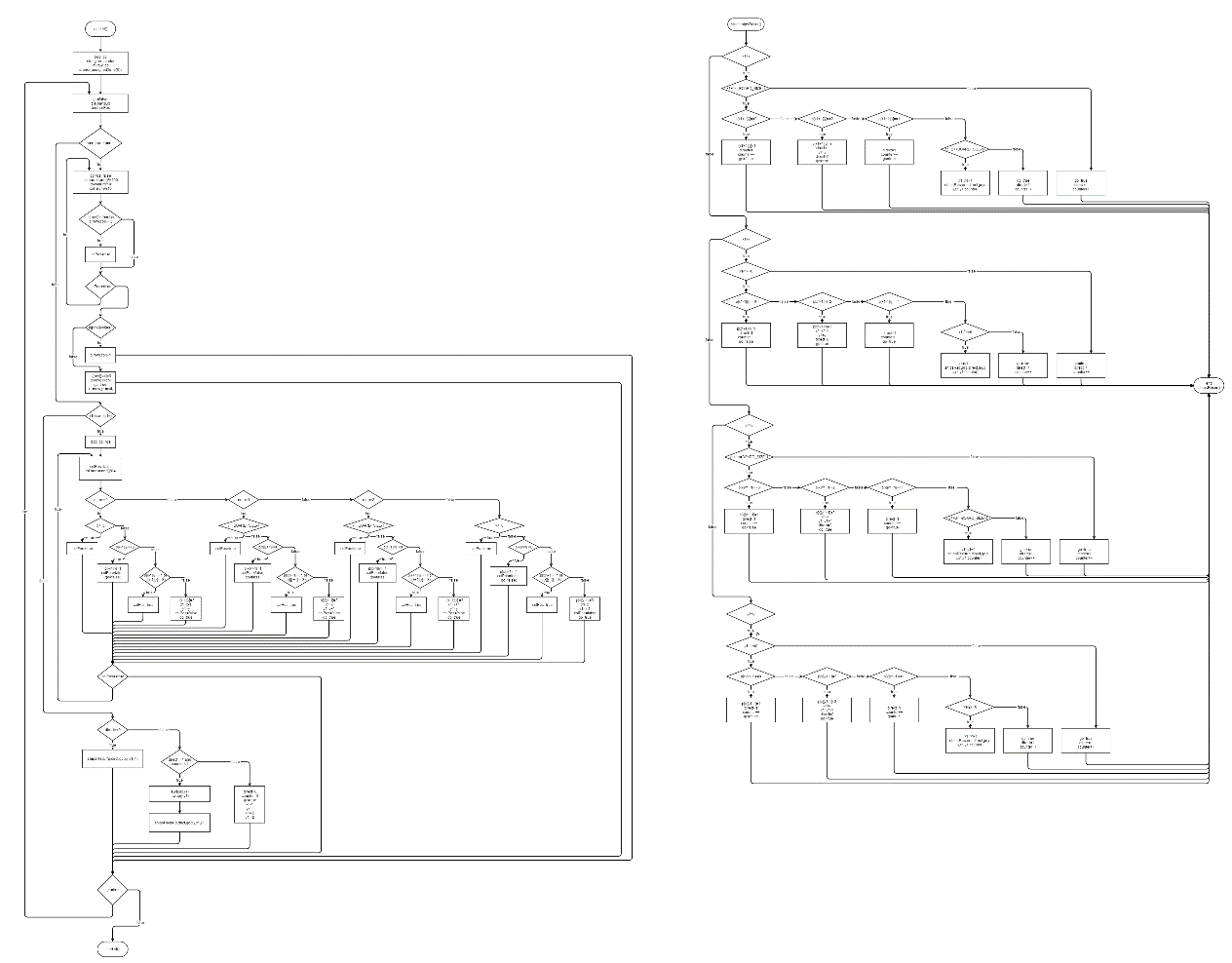
}

} while(hit again)

}

**4.Flowchart**

****



1. **code**

#ifndef SHIP\_H

#define SHIP\_H

class Ship {

private:

int size;

char direct;

int x;

int y;

public:

void setDirection(char);

void setCoPosition(int, int);

void setSize(int);

int getSize()const;

int getX()const;

int getY()const;

char getDirect()const;

};

void Ship::setCoPosition(int a, int b) {

x = a;

y = b;

}

void Ship::setDirection(char d) {

direct = d;

}

void Ship::setSize(int n) {

size = n;

}

int Ship::getX()const {

return x;

}

int Ship::getY()const {

return y;

}

char Ship::getDirect() const {

return direct;

}

int Ship::getSize()const {

return size;

}

#endif /\* SHIP\_H \*/

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*

\* File: ShipB.h

\* Author: Administrator

\*

\* Created on April 11, 2016, 12:27 PM

\*/

#include"Ship.h"

#include"BombS.h"

#ifndef SHIPB\_H

#define SHIPB\_H

#include <vector>

#include<iostream>

using namespace std;

class BattleShip : public Ship {

private:

friend class Bomb; //friend class

bool sink;

int damagePart;

vector<pair< int, int> >shipLoca; //store the x and y 's value

public:

void clearShipLoca(); //clear the wrong points,which do not appear on the board

void setWholeShip(); //draw the ship on the board

void setSink(bool);

bool isSink() const;

void setDamage(int);

int getDamage()const;

vector<pair< int, int > >getShipLoca()const;

};

void BattleShip::clearShipLoca() {

shipLoca.clear();

}

vector<pair< int, int > > BattleShip::getShipLoca()const {

return shipLoca;

}

void BattleShip::setWholeShip() {

int size = getSize();

int x = getX();

int y = getY();

char d = getDirect();

if (d == 'n') {

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

shipLoca.push\_back(make\_pair(x + i, y));

}

}

if (d == 's') {

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

shipLoca.push\_back(make\_pair(x - i, y));

}

}

if (d == 'w') {

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

shipLoca.push\_back(make\_pair(x, y + i));

}

}

if (d == 'e') {

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

shipLoca.push\_back(make\_pair(x, y - i));

}

}

}

void BattleShip::setDamage(int n) {

damagePart = n;

}

void BattleShip::setSink(bool y) {

sink = y;

}

bool BattleShip::isSink() const {

return sink;

}

int BattleShip::getDamage() const {

return damagePart;

}

#endif /\* SHIPB\_H \*/

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*

\* File: Bomb.h

\* Author: Administrator

\*

\* Created on April 11, 2016, 11:35 AM

\*/

#include "Ship.h"

#ifndef BOMB\_H

#define BOMB\_H

class Bomb {

public:

virtual int power(Ship s) = 0;

virtual int getNumber() const = 0;

};

#endif /\* BOMB\_H \*/

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*

\* File: BombN.h

\* Author: Administrator

\*

\* Created on April 11, 2016, 11:38 AM

\*/

#include "Bomb.h"

#ifndef BOMBN\_H

#define BOMBN\_H

class BombN : public Bomb {

protected:

int number;

public:

BombN() {

number = 100;

};

virtual int power(Ship s) {

return 1;

}

void setNumber();

virtual int getNumber()const;

};

int BombN::getNumber()const {

return number;

}

void BombN::setNumber() {

number--;

}

#endif /\* BOMBN\_H \*/

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*

\* File: BombS.h

\* Author: Administrator

\*

\* Created on April 11, 2016, 1:18 PM

\*/

//#include "BombN.h"

#include "Bomb.h"

#ifndef BOMBS\_H

#define BOMBS\_H

class BombS : public Bomb {

// private;

// int damage;

int number;

public:

BombS() {

number = 1;

};

virtual int power(Ship s) {

return s.getSize();

}

void setNumber();

virtual int getNumber()const;

};

int BombS::getNumber()const {

return number;

}

void BombS::setNumber() {

number--;

}

//int BombS::bombLeft() {

// return number;

//}

//

//void BombS::setNumber() {

// if (number >= 1)

// number--;

// else

// cout << "no bomb " << endl;

//}

#endif /\* BOMBS\_H \*/

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*

\* File: main.cpp

\* Author: Administrator

\*

\* Created on April 10, 2016, 5:23 PM

\*/

#include <cstdlib>

#include <iostream>

//#include "Ship.h"

#include "ShipB.h"

#include "BombN.h"

#include "BombS.h"

#include <fstream>

#include<cstring>

using namespace std;

const int BOARD\_SIZE = 10; //the board size

const int FLEET\_ARRAY = 5; //fleet size

struct Info {

char name[100];

bool win;

int steps;

};

void fillArray(int \*\*&p, int\*\*&c, int size); //create a board

void play(int\*\*&c, int\*\*p, BombN&B1, BombS&B2, BattleShip\*\*AIShip, int&); //player

void AI(int\*\*&p, int\*\*c, BombN&B1, BombS&B2, BattleShip\*\*playShip, int&, int&, int&, int&, int&, int&, int&); //computer

void realBoard(int\*\*p, int\*\*c); //the system's board;

void display(int\*\*p, int\*\*c);

void set(int\*\*&p, int x, int y, int z); //make change on the board

void playArrange(int\*\*&, int\*\*&, BattleShip\*\*&); //how to replace player's ships

void AIArrange(int\*\*&, int \*\*&, BattleShip\*\*&); //replace ai's ships

void pShip(BattleShip\*\*&playShip); //player's ships set

void cShip(BattleShip\*\*&AIShip); //ai's ships set

bool outBoard(int, int);

bool isWin(int\*\*p); //if no 2 on the board one player win;

int findShipOrder(BattleShip\*\* ship, int x, int y); //find the ship number to recorder the ship damage condition

void shipsRecur(int\*\*&p, int&, bool&, int&x, int&y, int&x1, int&y1, int&);

/\*1 - put the shell there

\* 0 - no shell there

\* 2 - the ship stay there

\* 3 - boat be hit

\*/

int main(int argc, char\*\* argv) {

cout << "BATTLE SHIP" << endl;

Info recorder, highScore;

string remainder;

int answer;

bool go, game;

// getline(cin,remainder);

// strcpy(recorder.name,remainder.c\_str());

do {

game = false;

do {

go = false;

cout << "1.New Game" << endl;

cout << "2.High score" << endl;

cout << "3 End" << endl;

cout << "Choose 1 to begin new game, choose 2 to display the high score" << endl;

cin>>answer;

if (cin.fail()) {

cout << "invalid input" << endl;

cin.clear();

getline(cin, remainder);

go = true;

}

if (answer < 1 || answer > 3) {

cout << "input out of range" << endl;

go = true;

}

} while (go);

getline(cin, remainder);

if (answer == 1) {

cout << "type your name" << endl;

string name;

getline(cin, name);

strcpy(recorder.name, name.c\_str());

} else if (answer == 2) {

game = true;

fstream f;

f.open("save.txt", ios::in | ios::binary);

f.seekg(0, ios::end);

streampos ps = f.tellg();

if (ps == 0) {

f.close();

cout << "No High score recorder" << endl;

} else {

f.seekg(ios::beg);

f.read(reinterpret\_cast<char\*> (&highScore), sizeof (highScore));

f.close();

cout << "High score: " << endl;

cout << "Name: " << highScore.name << endl;

if (highScore.win)

cout << "win this game at the ";

else

cout << "loose this game at the ";

cout << highScore.steps << " steps" << endl;

}

} else {

return 0;

}

} while (game);

int step = 0; //record the step's number

int whoWin = 0; //1: player win,2:ai win

int \*\*p = new int\*[BOARD\_SIZE]; //player's board

int \*\*c = new int\*[BOARD\_SIZE]; //computer's board,computer put shell on player's board

int directTurn = 0; //when value==1,change shell direction, when value==2,need seek other ship

int counter = 0; //记录方向变化的次数

int pX = -1, pY = -1; //for Ai -recorder the first location of ship was hit

int x1 = -2, y1 = -2; //the previous shell location

BombN B1;

BombS B2;

fillArray(p, c, BOARD\_SIZE);

BattleShip \*\*playShip = new BattleShip\*[FLEET\_ARRAY];

pShip(playShip);

BattleShip \*\*AIShip = new BattleShip\*[FLEET\_ARRAY];

cShip(AIShip);

playArrange(p, c, playShip);

AIArrange(c, p, AIShip);

// realBoard(p, c); //check

/\*determine who win the game\*/

bool turn = true;

do {

cout << endl << endl;

cout << "0-unknown point" << endl;

cout << "1-shell is there(no ship)" << endl;

cout << "2-ship is there" << endl;

cout << "3-ship is hit" << endl;

cout << "Player turn: " << endl;

step++;

play(c, p, B1, B2, AIShip, whoWin);

if (whoWin == 1 || whoWin == 2) {

break;

// cout<<"break 1"<<endl;

}

cout << endl << endl;

cout << "Computer turn: " << endl;

AI(p, c, B1, B2, playShip, pX, pY, x1, y1, directTurn, counter, whoWin);

if (whoWin == 1 || whoWin == 2)

break;

} while (turn);

realBoard(p, c);

if (whoWin == 1) {

recorder.win = true;

} else

recorder.win = false;

recorder.steps = step;

fstream fs;

fs.open("save.txt", ios::in | ios::out | ios::binary);

fs.seekg(0, ios::end);

streampos ps2 = fs.tellg();

if (ps2 == 0) {

fs.write(reinterpret\_cast<char\*> (&recorder), sizeof (recorder));

fs.close();

} else {

fs.seekg(ios::beg);

fs.read(reinterpret\_cast<char\*> (&highScore), sizeof (highScore));

fs.seekp(ios::beg);

if (recorder.win&&!highScore.win) {

fs.write(reinterpret\_cast<char\*> (&recorder), sizeof (recorder));

} else if (!recorder.win&&!highScore.win) {

if (recorder.steps > highScore.steps)

fs.write(reinterpret\_cast<char\*> (&recorder), sizeof (recorder));

} else if (recorder.win && highScore.win) {

if (recorder.steps < highScore.steps)

fs.write(reinterpret\_cast<char\*> (&recorder), sizeof (recorder));

}

fs.close();

}

return 0;

}

int findShipOrder(BattleShip\*\*ship, int x, int y) {

// cout << "test point 4" << endl;

for (int i = 0; i < FLEET\_ARRAY; i++) {

// cout << "x is " << x << "; " << "y is " << y << endl;

int size = ship[i]->getSize(); //each ship's length

// int len;//record the ship's len;

vector<pair<int, int> >v;

// find the ship's all location

// cout << "test point 5" << endl;

// for (int k = 0; k < size; k++) {//find the position of the ship's grids

// v.push\_back(make\_pair(ship[i]->getShipLoca()[k].first, ship[i]->getShipLoca()[k].second));

// }

// cout << "test point 6" << endl;

bool hit = false;

for (int j = 0; j < ship[i]->getShipLoca().size(); j++) {

if (ship[i]->getShipLoca()[j].first == x && ship[i]->getShipLoca()[j].second == y) {

hit = true; //find the ship which was hit

// break;

}

}

// cout << "test poitn 6" << endl;

if (hit) {//if hit

return i; //return the ship order number

}

}

return -1;

}

bool isWin(int\*\*p) {

for (int i = 0; i < BOARD\_SIZE; i++) {

for (int j = 0; j < BOARD\_SIZE; j++) {

if (p[i][j] == 2)

return false;

}

}

return true;

}

/\*将5条传生成object后放在一个数组里\*/

void pShip(BattleShip \*\*&playShip) {//create player's ship array

for (int i = 0; i < FLEET\_ARRAY; i++) {

playShip[i] = new BattleShip();

}

}

/\*将5条传生成object后放在一个数组里\*/

void cShip(BattleShip \*\*&AIShip) {//create Ai's ship array

for (int i = 0; i < FLEET\_ARRAY; i++) {

AIShip[i] = new BattleShip();

}

}

/\*player's 5 ships put on the player's board\*/

void playArrange(int\*\*&p, int\*\*&c, BattleShip\*\*&ship) {

for (int i = 0; i < FLEET\_ARRAY; i++) {

string remainder;

int x, y;

char d;

bool go = false;

cout << "ship " << i + 1 << " size is " << i + 1 << endl;

do {

go = false;

cout << "type the ship bow position (x y) 0-9 " << endl;

cin >>x;

// cin.get();

cin>>y;

if (x > 9 || x < 0 || y > 9 || y < 0) {

cout << "invalid input" << endl;

go = true;

}

if (cin.fail()) {

cin.clear();

getline(cin, remainder);

cout << "invalid input" << endl;

go = true;

}

cout << "type the direction of the ship bow (n,s,e,w)" << endl;

cin>>d;

if (toupper(d) != 'N' && toupper(d) != 'S' && toupper(d) != 'E' && toupper(d) != 'W') {

// cout << "test point 1" << endl;

// cout<<"wrong input"<<endl;

cout << "invalid input" << endl;

go = true;

}

ship[i]->setSize(i + 1);

ship[i]->setDirection(d);

ship[i]->setCoPosition(x, y);

ship[i]->setWholeShip();

vector<pair<int, int> >v = ship[i]->getShipLoca();

for (int j = 0; j < v.size(); j++) {

if (outBoard(v[j].first, v[j].second)) {

cout << "part of the boat out of the board" << endl;

go = true;

v.clear();

ship[i]->clearShipLoca();

}

}

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

if (p[v[k].first][v[k].second] == 2) {

cout << "there is another boat there" << endl;

// cout<<"there boat there"<<endl;

ship[k]->clearShipLoca();

go = true;

v.clear();

}

}

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

p[v[i].first][v[i].second] = 2;

}

getline(cin, remainder);

} while (go);

realBoard(p, c);

}

}

bool outBoard(int a, int b) {

if (a > 9 || a < 0 || b > 9 || b < 0)

return true;

else

return false;

}

/\*place 5 ships of computer on the computer board\*/

void AIArrange(int\*\*&c, int\*\*&p, BattleShip\*\*&ship) {

srand(unsigned(time(0)));

for (int i = 0; i < FLEET\_ARRAY; i++) {

// cout << "i is " << i << endl;

bool goAI = false;

do {

goAI = false;

int index = rand() % 100;

int row = index / 10;

int col = index % 10;

if (c[row][col] == 2)

goAI = true;

if (goAI) {

} else {

// cout << "x is " << row << " ; " << " y is " << col << endl;

ship[i]->setCoPosition(row, col);

int dirct = rand() % 4;

// cout << "direction is " << dirct << endl;

int size = i + 1;

ship[i]->setSize(size);

if (dirct == 0)

ship[i]->setDirection('e');

else if (dirct == 1)

ship[i]->setDirection('s');

else if (dirct == 2)

ship[i]->setDirection('w');

else

ship[i]->setDirection('n');

ship[i]->setWholeShip();

// cout << "ship is at: ";

// for (int j = 0; j < size; j++)

// cout << "(" << ship[i]->getShipLoca()[j].first << " " << ship[i]->getShipLoca()[j].second << ")";

// cout << endl;

for (int j = 0; j < ship[i]->getShipLoca().size(); j++) {

if (outBoard(ship[i]->getShipLoca()[j].first, ship[i]->getShipLoca()[j].second)) {

// cout << "point 1" << endl;

goAI = true;

// v1.clear();

ship[i]->clearShipLoca();

break;

}

}

for (int j = 0; j < ship[i]->getShipLoca().size(); j++) {

if (c[ship[i]->getShipLoca()[j].first][ship[i]->getShipLoca()[j].second] == 2) {

// cout << "point 2" << endl;

ship[i]->clearShipLoca();

goAI = true;

// v1.clear();

break;

}

}

for (int j = 0; j < ship[i]->getShipLoca().size(); j++) {

// cout << "point 3" << endl;

c[ship[i]->getShipLoca()[j].first][ship[i]->getShipLoca()[j].second] = 2;

}

// ship[i]->clearShipLoca();

// cout << "test ~~~~~~~~~~~~~~1" << endl;

}

} while (goAI);

}

}

void play(int\*\*&c, int \*\*p, BombN&B1, BombS&B2, BattleShip\*\*AIShip, int&whoWin) {

bool go;

string remainder;

do {

go = false;

int x, y, s;

bool invalid;

/\*valid input\*/

do {

invalid = false;

cout << "Special shell number: " << B2.getNumber() << endl;

cout << "Type 0 for normal shell, type 1 for special shell" << endl;

cout << "Choose a coordination location and shell type (x y z),eg(1 5 0) " << endl;

cin >> x >> y>>s;

// cout<<"s is "<<s<<" ; special shell number is "<<B2.getNumber()<<endl;

if (s == 1 && B2.getNumber() < 1) {

cout << "no more special shell" << endl;

invalid = true;

} else {

if (x > 9 || x < 0 || y > 9 || y < 0 || s < 0 || s > 1) {

cout << "invalid input" << endl;

invalid = true;

}

if (cin.fail()) {

cin.clear();

getline(cin, remainder);

cout << "invalid input" << endl;

invalid = true;

}

}

} while (invalid);

// cout << "test point 2" << endl;

/\*change the data of the class bomb\*/

if (s == 0)

B1.setNumber(); //normal shell number --;

else B2.setNumber(); //special shell number --;

if (c[x][y] == 0) {//mark the shell ,no ship there

c[x][y] = 1;

}

if (c[x][y] == 2) {//hit boat

go = true;

if (s == 0) {

c[x][y] = 3; //normal bomb just mark one position

} else {

// cout << "test point 3" << endl;

//test the ship one by one to determine whether it is hit

int index = findShipOrder(AIShip, x, y);

cout << "index is " << index << endl;

for (int i = 0; i < index + 1; i++) {

c[AIShip[index]->getShipLoca()[i].first][AIShip[index]->getShipLoca()[i].second] = 3;

}

}

}

display(p, c);

if (isWin(c)) {

cout << "Player win" << endl;

whoWin = 1;

go = false;

}

} while (go);

}

void AI(int\*\*&p, int\*\*c, BombN&B1, BombS&B2, BattleShip\*\*playShip, int& x,

int& y, int&x1, int&y1, int&direct, int &counter, int&whoWin) {

bool go;

string remainder;

int row, col;

srand((unsigned) time(0));

do {

go = false;

bool coPos;

if (x == -1 && y == -1) {//no ship be found

do {//这个点已经走过

coPos = false;

int num = rand() % 100;

row = num / 10;

col = num % 10;

if (p[row][col] == 1 || 3 == p[row][col]) {

coPos = true;

}

} while (coPos);

if (p[row][col] == 0) {//

p[row][col] = 1;

} else {

p[row][col] = 3;

x = row;

y = col; //find the ship

go = true;

x1 = row;

y1 = col;

}

} else {//find ship

if (x1 == x && y1 == y) {

// 只有一节被打中

bool colPos;

// int num = -1;//什么用

do {

// num++;//什么用

colPos = false;

int num = rand() % 4;

if (num == 0) {//ship bow toward south

if (x <= 0)

colPos = true;

else {

if (p[x - 1][y] == 0) {//no ship there

p[x - 1][y] = 1;

colPos = false;

go = false;

} else if (p[x - 1][y] == 1 || p[x - 1][y] == 3) {

colPos = true;

} else {

p[x - 1][y] = 3;

x1 = x - 1; //记录命中的坐标

y1 = y;

colPos = false;

go = true;

}

}

} else if (num == 1) {//ship bow toward west

if (y == BOARD\_SIZE - 1)

colPos = true;

else {

if (p[x][y + 1] == 0) {//no ship there

p[x][y + 1] = 1;

colPos = false;

go = false;

} else if (p[x][y + 1] == 1 || p[x][y + 1] == 3) {

colPos = true;

} else {

p[x][y + 1] = 3;

x1 = x;

y1 = y + 1;

colPos = false;

go = true;

}

}

} else if (num == 2) {//ship bow towards north

if (x == BOARD\_SIZE - 1)

colPos = true;

else {

if (p[x + 1][y] == 0) {//no ship there

p[x + 1][y] = 1;

colPos = false;

go = false;

} else if (p[x + 1][y] == 1 || p[x + 1][y] == 3) {

colPos = true;

} else {

p[x + 1][y] = 3;

x1 = x + 1;

y1 = y;

colPos = false;

go = true;

}

}

} else {//ship bow toward east

if (y <= 0)

colPos = true;

else {

if (p[x][y - 1] == 0) {//no ship there

p[x ][y - 1] = 1;

colPos = false;

go = false;

} else if (p[x][y - 1] == 1 || p[x][y - 1] == 3) {

colPos = true;

} else {

p[x][y - 1] = 3;

x1 = x;

y1 = y - 1;

colPos = false;

go = true;

}

}

}

// else {

// x = -1;

// y = -1;

// x1 = -2;

// y1 = -2;

// colPos = false;

// go = false;

// }

} while (colPos);

} else {//最少打中两节

if (direct == 0) {//same direct

shipsRecur(p, direct, go, x, y, x1, y1, counter);

} else if (direct == 1 && counter < 2) {//counter=2 意味着转了两次方向，已知点的两头

// 都已经试探过了

swap(x, x1);

swap(y, y1);

shipsRecur(p, direct, go, x, y, x1, y1, counter);

} else {

direct = 0;

counter = 0;

go = true;

x = -1;

y = -1;

x1 = -2;

y1 = -2;

}

}

}

display(p, c);

if (isWin(p)) {

cout << "AI win" << endl;

whoWin = 2;

break;

}

} while (go);

}

void shipsRecur(int\*\*&p, int&direct, bool&go, int&x, int&y, int&x1, int&y1, int&counter) {

//same direct

if (x1 > x) {

if (x1 + 1 <= BOARD\_SIZE - 1) {//x1 toward south do not touch the boundary

if (p[x1 + 1][y] == 0) {//没击中

p[x1 + 1][y] = 1;

direct = 1; // 未击中，转向

counter++; //专项次数加1

go = false;

} else if (p[x1 + 1][y] == 2) {

p[x1 + 1][y] = 3;

x1 = x1 + 1;

y1 = y;

direct = 0; //击中，不用转向

go = true;

} else if (p[x1 + 1][y] == 1) {

direct = 1;

counter++; //该点已经试探过，转向，且转向次数+1

go = true;

} else {//two ships together,need to jump of the 3

if (x1 + 2 <= BOARD\_SIZE - 1) { //该点显示击中，可能两船交叉，跳过该点，判断是否为边界

x1 = x1 + 1;

shipsRecur(p, direct, go, x, y, x1, y1, counter);

} else {//如果是边界，转向

go = true;

direct = 1;

counter++;

}

}

} else {

direct = 1;

counter++;

go = true;

}

}

if (x1 < x) {

if (x1 - 1 >= 0) {//x1 toward south do not touch the boundary

if (p[x1 - 1][y] == 0) {//没击中

p[x1 - 1][y] = 1;

direct = 1;

counter++;

go = false;

} else if (p[x1 - 1][y] == 2) {

p[x1 - 1][y] = 3;

x1 = x1 - 1;

y1 = y;

direct = 0;

go = true;

} else if (p[x1 - 1][y] == 1) {

direct = 1;

counter++;

go = true;

} else {//two ships together,need to jump of the 3

if (x1 - 2 >= 0) {

x1 = x1 - 1;

shipsRecur(p, direct, go, x, y, x1, y1, counter);

} else {

go = true;

direct = 1;

counter++;

}

}

} else {

direct = 1;

counter++;

go = true;

}

}

if (y1 > y) {

if (y1 + 1 <= BOARD\_SIZE - 1) {//x1 toward south do not touch the boundary

if (p[x][y1 + 1] == 0) {//没击中

p[x][y1 + 1] = 1;

direct = 1;

counter++;

go = false;

} else if (p[x][y1 + 1] == 2) {

p[x][y1 + 1] = 3;

x1 = x;

y1 = y1 + 1;

direct = 0;

go = true;

} else if (p[x][y1 + 1] == 1) {

direct = 1;

counter++;

go = true;

} else {//two ships together,need to jump of the 3

if (y1 + 2 <= BOARD\_SIZE - 1) {

y1 = y1 + 1;

shipsRecur(p, direct, go, x, y, x1, y1, counter);

} else {

go = true;

direct = 1;

counter++;

}

}

} else {

direct = 1;

counter++;

go = true;

}

}

if (y1 < y) {

if (y1 - 1 >= 0) {//x1 toward south do not touch the boundary

if (p[x][y1 - 1] == 0) {//没击中

p[x][y1 - 1] = 1;

direct = 1;

counter++;

go = false;

} else if (p[x][y1 - 1] == 2) {

p[x][y1 - 1] = 3;

x1 = x;

y1 = y1 - 1;

direct = 0;

go = true;

} else if (p[x][y1 - 1] == 1) {

direct = 1;

counter++;

go = true;

} else {//two ships together,need to jump of the 3

if (y1 - 2 >= 0) {

y1 = y1 - 1;

shipsRecur(p, direct, go, x, y, x1, y1, counter);

} else {

go = true;

direct = 1;

counter++;

}

}

} else {

direct = 1;

counter++;

go = true;

}

}

}

void set(int\*\*p, int x, int y, int z) {

if (p[x][y] == 0 && p[x][y] != 2) {//no ship on this position

p[x][y] = 1;

}

if (p[x][y] == 0 && p[x][y] == 2 && z == 1) {//special shell and ship be hit

p[x][y] == 2;

}

}

void fillArray(int\*\*&p, int\*\*&c, int size) {

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

p[i] = new int[size] {

0

};

c[i] = new int[size] {

0

};

}

}

void realBoard(int \*\*p, int\*\*c) {

cout << "Player Computer" << endl;

cout << " 0 1 2 3 4 5 6 7 8 9" << '\t' << " 0 1 2 3 4 5 6 7 8 9" << endl;

cout << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << '\t' << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

for (int i = 0; i < BOARD\_SIZE; i++) {

cout << i << "| ";

for (int j = 0; j < BOARD\_SIZE; j++)

cout << p[i][j] << " ";

cout << '\t' << i << '|';

for (int j = 0; j < BOARD\_SIZE; j++)

cout << c[i][j] << " ";

cout << endl;

}

}

void display(int \*\*p, int\*\*c) {

cout << "Player Computer" << endl;

cout << " 0 1 2 3 4 5 6 7 8 9" << '\t' << " 0 1 2 3 4 5 6 7 8 9" << endl;

cout << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << '\t' << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

for (int i = 0; i < BOARD\_SIZE; i++) {

cout << i << "| ";

for (int j = 0; j < BOARD\_SIZE; j++)

cout << p[i][j] << " ";

cout << '\t' << i << '|';

for (int j = 0; j < BOARD\_SIZE; j++) {

if (c[i][j] == 2)

cout << 0 << " ";

else

cout << c[i][j] << " ";

}

cout << endl;

}

}