Battleship

Project2

CSC- 17A

Valentinno Cruz

06 – August – 2021a

Content

1.Introduction………………………………………….3-4  
Rule and Gameplay  
Thoughts after Program

2. Development………………………..………………4-5  
Approach Strategy

3. Variables list…………………………….……..……5-6

4. Topic Covered (Checklist)…………………….…....7-8

5. Libraries included……………………………………..8

6. Pseudo Code……………………………………….8-10

7. Flowchart…………………………………………11-15

8. Code……………………………………………...16-55

1. Introduction

**Rules and Gameplay**

The players can choose the size of the table which 8x8 to 10x10. After the player chooses the size, a template class object is created, and the object will create two game class objects which will create two tables, fake and real, for each of them. One of the game object’s tables are for player, the others are for AI. Both of them have 5 ships which are 1 5-unit ship, 1 4-unit ship, 1 3-unit ship, and 2 2-unit ships. After the player enter the coordinate to place the ship, the table will be refresh and use 2 to 5 to label the ships. The coordinates of AI’s ships will be place randomly when the AI game’s class is created. Player needs to hit all the ships to win the game. In the game, “X” means hit and “O” means miss. Every time after check the validation, the program will scan the table to check whether there is any numbers which is ships on the table. If there isn’t any numbers on the table, the game ends.

**Thoughts after Program**

The game seems very simple, but the AI’s fire part is very complicated because it is very difficult to make an AI acts like a human player. I want the AI check the coordinates around the hit coordinate, and keep fire when it gets the second hit. If one side is “O” or the side of the table, the AI needs to check the other side too. This needs many Boolean variables. Also, the AI table shown to the player isn’t the real table, it is a clear table and after the player fire, the program will compare the coordinate to the real table. Then it records and shows “O”, “X”, or invalid input. The validation part costs me a lot of time too because I use string as an input type and input in A1 form to let the player input the coordinates. This can check the length easily, but I need to use ascii code to translate after check the length. It is possible to make the Ai smarter which is divided the table into several sections and randomly fire each of the section to increase the accuracy, but it needs more codes and better logic.

2. Development

Approach Strategy

The battleship needs at least three tables, one for player and two for AI. It is too difficult to use one-dimension arrays. It is easier to use 2-dimension arrays. Also, I use A-J to label the rows and 0-9 to label the columns. It makes the players enter the coordinates clearly and prevent them get confused. It is possible the tables which are larger than 10x10, but there are only 0-9 for digits, and it is not a good idea to use low case characters with digits because it is confusing. I have tried to let the player to choose which ship they want to place first, but there are 2 2-unit ships, so I need to use a Boolean to remember the first 2-unit ship. However, there are many bugs and I couldn’t fix it. Therefore, I let the player place the 5-unit ship, then 4-unit ship, and so on. After the player’s place ship part, I need to random the AI ships’ coordinates. Because I use an array to store the ship units, so I can avoid the oversize by subtract the units such as srand()%num-5(num is the size of the table). After generate the coordinate, the program will random to place it horizontally or vertically. If the ship overlaps, it will try to place it in other way. If it is still invalid, the program will random the coordinates again.

After the preparing, I use a switch to separate the player’s fire turn and AI’s fire turn. If the game isn’t over, the program to go to AI’s turn and so on. If the game ends in player’s turn, the program will jump to other case same as AIs. Also, I put a do-while loop outside the switch and repeat until the game is over.

For the AI’s fire part, I let the AI to fire randomly until it hits. After AI hits, the program will record the coordinate and check the four coordinates beside it until it gets second hit. After a second hit, AI will fire that direction until it get miss, touch the side, or overlap. Then, it will fire the opposite side until miss, oversize, overlap again. After it finishes these steps, it will go back to random fire mode.

3. Variables list

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | **Variable Name** | **Description** | **File** | **Line** |
| int | num | size of 2d dynamic array | main | 41 |
|  | hx, hy, x, y | current, pre fire coordinates | AI.h | 20 |
|  | oppcombo | opposite side combo counter | AI.h | 22 |
|  | combo | combo counter | AI.h | 23 |
|  | x1, x2, y1, y2 | 2 coordinates to place ship | Game.h | 45 |
|  | unit[5] | units of the ships in an array | Game.h | 46 |
|  | count | place ship unit counter | Game.cpp | 186 |
|  | max, min |  | Game.cpp | 238 |
|  | count=0 | place ship unit counter | Game.cpp | 239 |
|  | hplan | hit plan after first hit (cross) | Game.cpp | 390 |
|  | num | size of the table | Table.h | 34 |
| float | hit, miss | hit miss counter | Game.h | 49 |
| char | cx, cy | char format of x, y | AI.h | 24 |
|  | row=i+65 (overload) | table coordinate indicator | Game.cpp | 40 |
|  | temp (player) | temp to store char form unit.txt | Game.cpp | 66 |
|  | temp (ai) | temp to store char form unit.txt | Game.cpp | 81 |
|  | row=i+65 (print) | table coordinate indicator | Game.cpp | 161 |
|  | \*\*fake | fake table | Table.h | 13 |
|  | \*\*real | real table | Table.h | 14 |
| string | numException | array size exception string | main | 67 |
|  | place | player place input type | Game.h | 51 |
|  | fire | player fire input type | main | 372 |
|  | temp | exception string 1 | Game.cpp | 225 |
|  | temp | exception string 2 | Game.cpp | 240 |
|  | fire | player fire input type | Game.cpp | 332 |
| bool | invalid | num validation | main | 42 |
|  | done | finish fire | AI.h | 16 |
|  | cross[4] | cross 4 boxes around hit | AI.h | 17 |
|  | crossdone | if true back to random | AI.h | 18 |
|  | goback | invalid back to random | AI.h | 19 |
|  | oneend | one side finish | AI.h | 25 |
|  | combohit | keep fire the same direction | AI.h | 26 |
|  | bhit | hit Boolean | AI.h | 27 |
|  | over | over Boolean | Game.h | 47 |
|  | valid | validation Boolean | Game.h | 48 |
|  | win | win Boolean | Game.h | 50 |
|  | valid | ai place validation | Game.cpp | 95 |
| fstream | io | i/o file in function intro | main | 72 |
|  | io | i/o file in Game(player) constructor | Game.cpp | 65 |
|  | io | i/o file in Game(AI) constructor | Game.cpp | 82 |
| time\_t | tstart, tend | delay display ai fire | AI.cpp | 23 |
| Table | \*t | struct table pointer | Table.h | 35 |
| Game | p(num) | player game object | Why.h | 32 |
|  | ai(num, ‘c’) | AI game object | Why.h | 33 |
| Why<test> | a(num) | class Why object with test type | main | 59 |

4. Topic Covered (Checklist)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Chapter** | **type** | **code** | **File** | **line** |
| Memory Allocation | char\*\* | t->fake = new char \*[num]; | Table.cpp | 14 |
|  |  | t->real = new char \*[num]; | Table.cpp | 15 |
|  |  | for(int i=0;i<num;i++){ | Table.cpp | 16 |
|  |  | t->fake[i] = new char[num]; | Table.cpp | 17 |
|  |  | t->real[i] = new char[num]; | Table.cpp | 18 |
|  |  | } | Table.cpp | 19 |
| exception function | void | throw numException; | main | 65-70 |
| try block |  | try{ | main | 49 |
|  |  | numTest(num); | main | 50 |
|  |  | } | main | 52 |
| catch block |  | catch(string numException){ | main | 53 |
|  |  | cout<<numException | main | 55 |
|  |  | } | main | 57 |
| Constructor | Game | Game(int num); | Game.h | 22 |
| Call another constructor |  | Game::Game(int num):Table(num) { | Game.cpp | 64 |
| virtual destructor |  | virtual ~Table(); | Table.h | 32 |
| Destructor |  | ~Game(); | Game.h | 33 |
| template |  | template <class T> | Why.h | 15 |
|  |  | t = new T(num); | Why.h | 20 |
| base class |  | class Table{ | Table.h | 16 |
| derived class  inheritance |  | class Game:public Table, public AI, public abstractGame{ | Game.h | 19 |
| abstract class |  | class abstractGame{ | Game.h | 15 |
|  |  | public: | Game.h | 16 |
| virtual function |  | virtual void check()=0; | Game.h | 17 |
|  |  | } | Game.h | 18 |
| delete 2d dynamic arr |  | for (int i=0;i<num;i++){ | Table.cpp | 28 |
|  |  | delete[] t->fake[i]; | Table.cpp | 29 |
|  |  | delete[] t->fake[i]; | Table.cpp | 30 |
|  |  | } | Table.cpp | 31 |
|  |  | delete[] t->fake; | Table.cpp | 32 |
|  |  | delete[] t->real; | Table.cpp | 33 |
| pointer notation |  | if(\*(\*(t->real+k)+x1)==' '){ | Game.cpp | 295 |
| class pointer | T | T \*t | Why.h | 27 |
| delete class pointer |  | delete t | Why.h | 24 |
| cctype | isdigit | isdigit(place[1]==0 || isdigit(place[3])==0)){ | Game.cpp | 232 |
| input binary | input | io.open("unit.txt", ios::in | ios::binary); | Game.cpp | 67 |
| output binary | output | io.open(“result.txt”, ios::out | ios::binary(; | Game.cpp | 326 |
| Array | int | int unit[5]; | Game.h | 46 |
| difftime |  | }while(difftime(end,start)<1); | AI.cpp | 27 |
| overload ostream |  | friend ostream & operator << (ostream &,const Game &); | Game.h | 32 |

5. Libraries included

* <cstdlib>
* <iostream>
* <ctime>
* <fstream>
* <iomanip>
* <cctype>
* Game.h
* AI.h
* Table.h
* Why.h

6. Pseudo Code

Ask the size of the table

check validation (range 8-10)

Create Why object

Inside Why object constructor

create test object and set pointer point to that object

inside test object

create 2 game objects (player, ai)

Reset table

ai generate ship when ai game object is created

Output table

do{

Input 2 coordinates to place ship

}while (invalid)

place other ship and check validation

**test object call game start{**

**do{**

Player enter coordinate to frie

check validation

check hit/miss and add count

display table again

check game over (no number s on the table)

if(over)

win=true

if(win=true)

player wins, game ends

else{

ai fire

check game over

}

if(over)

win=false

if(win=false)

player lose

game ends

}while(over==false);

**(AI fire)**

do{

if (not hit/combo)

random hit

if (hit)

check cross 4

if(all invalid)

go back to random

if(hit)

combo++

add count

else

add count

if (cross 4 coordinates hit)

continue fire that direction

if(invalid)

jump to next statement

oppcombo++

combo=0

if(miss)

oppcombo++

combo=0

add count

if(hit)

combo++, add count

if(oppcombo>0)

check the opposite side

if(invalid) go back to random

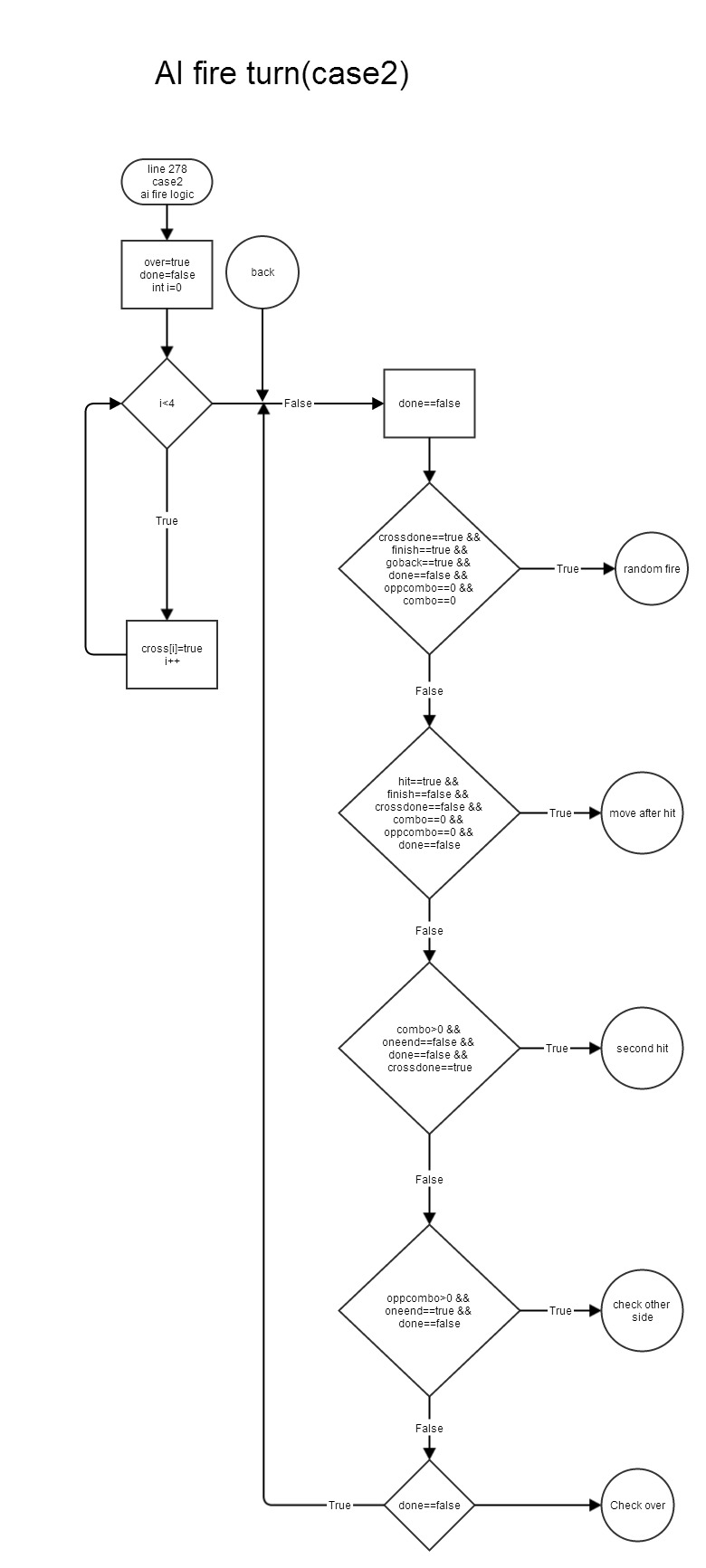
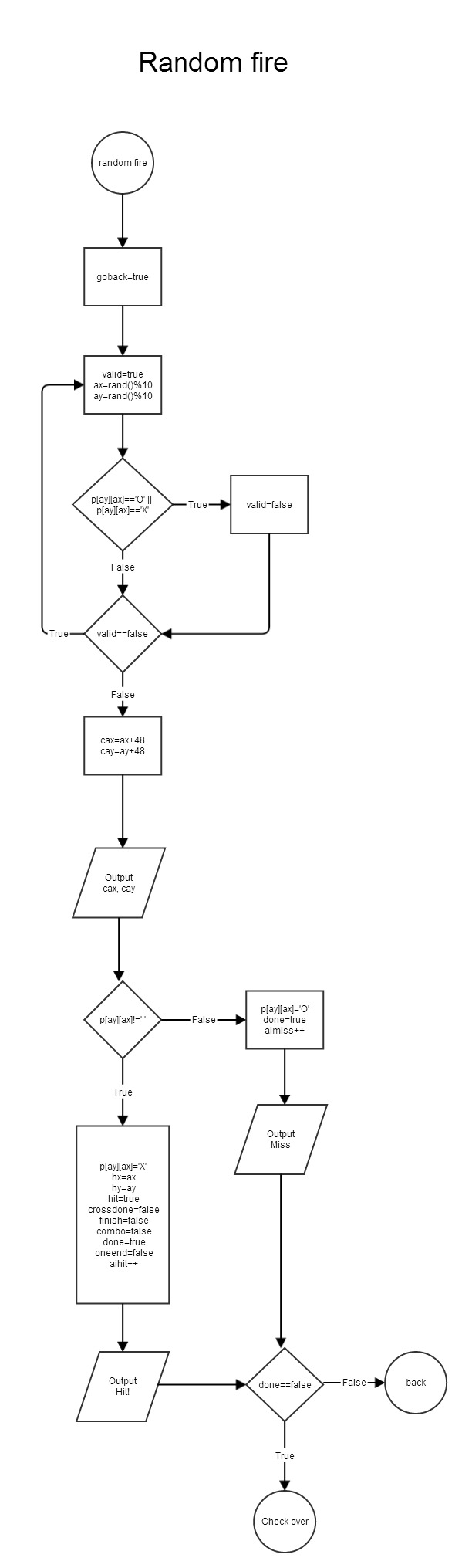
if(miss)

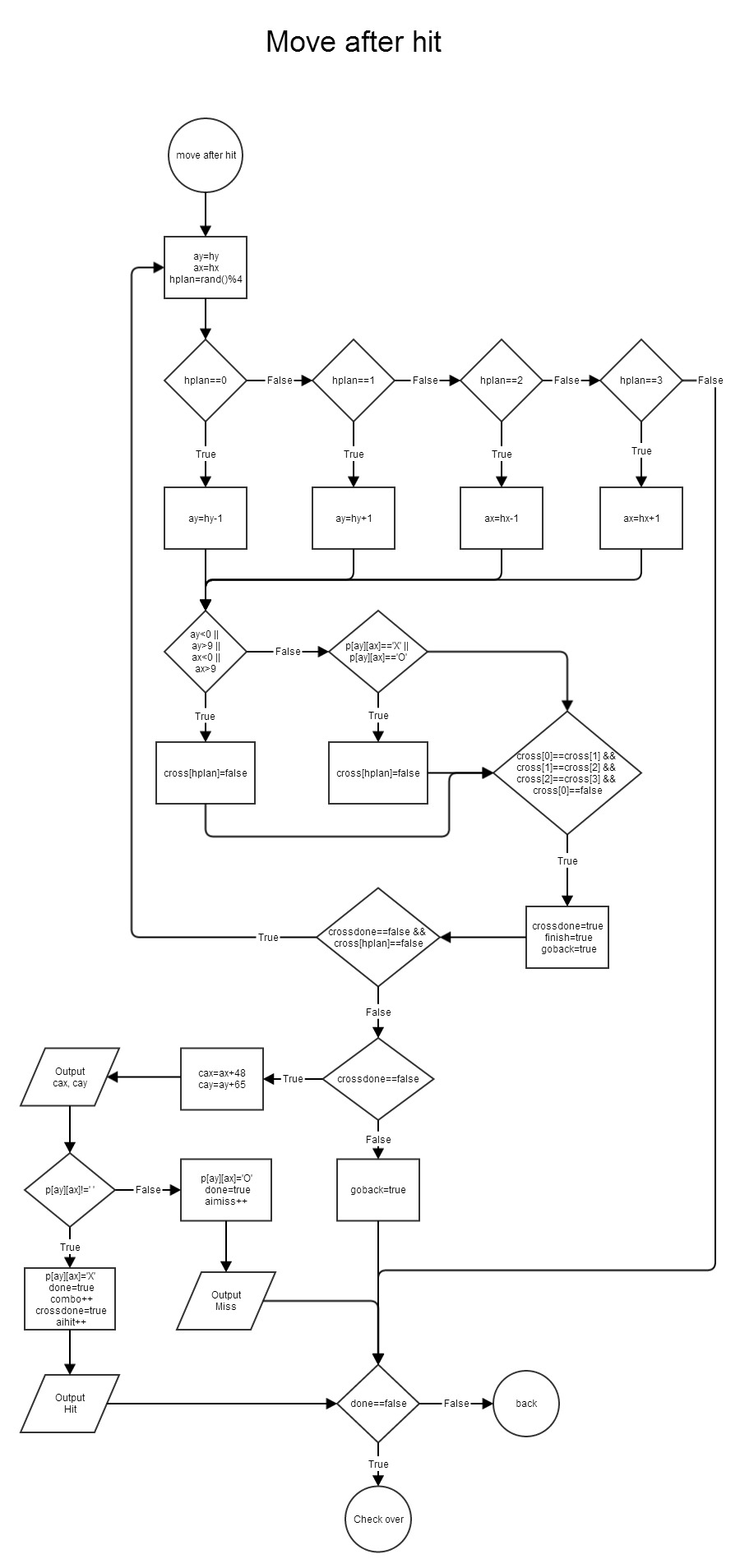
oppcombo=0, add count

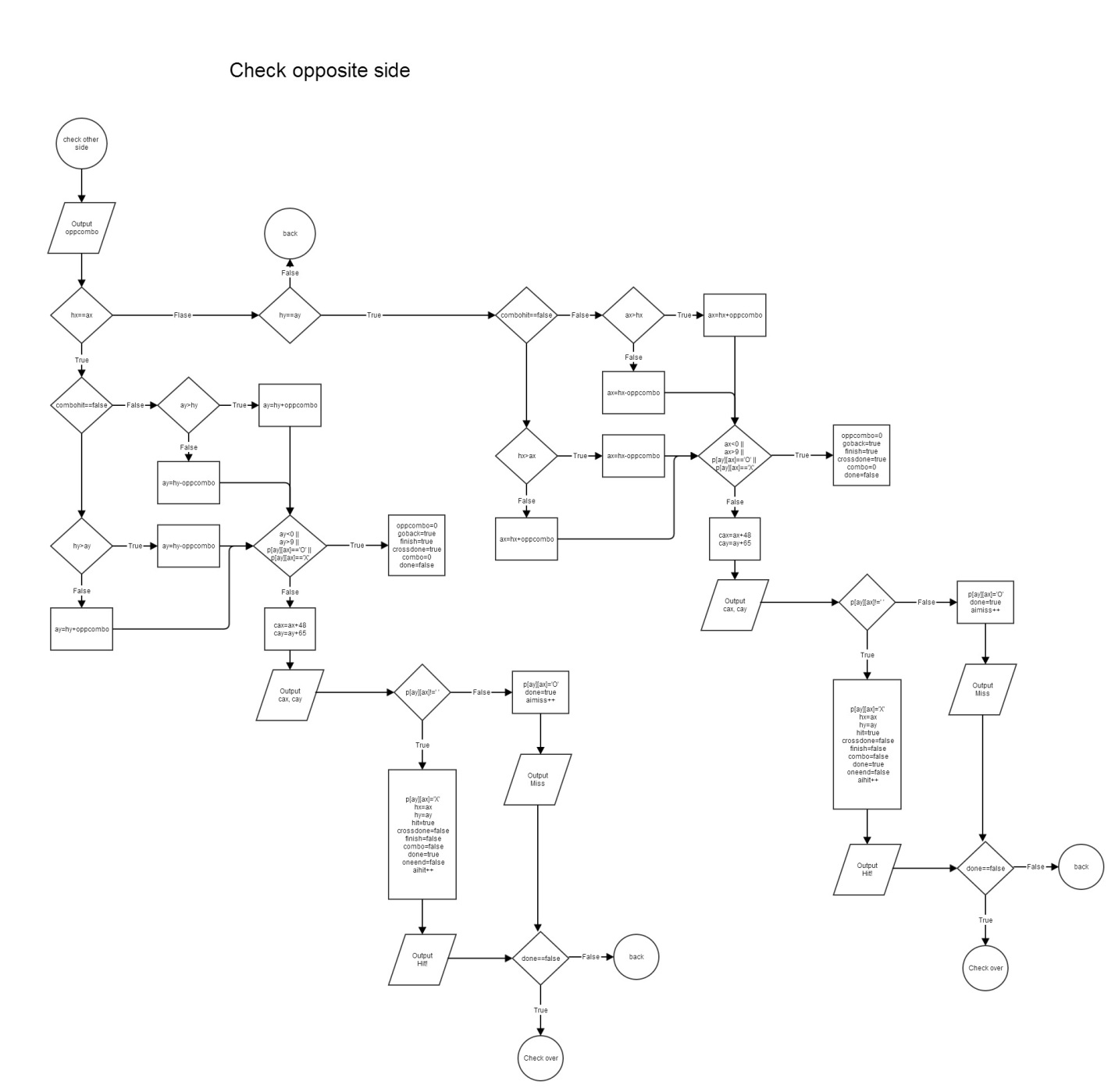
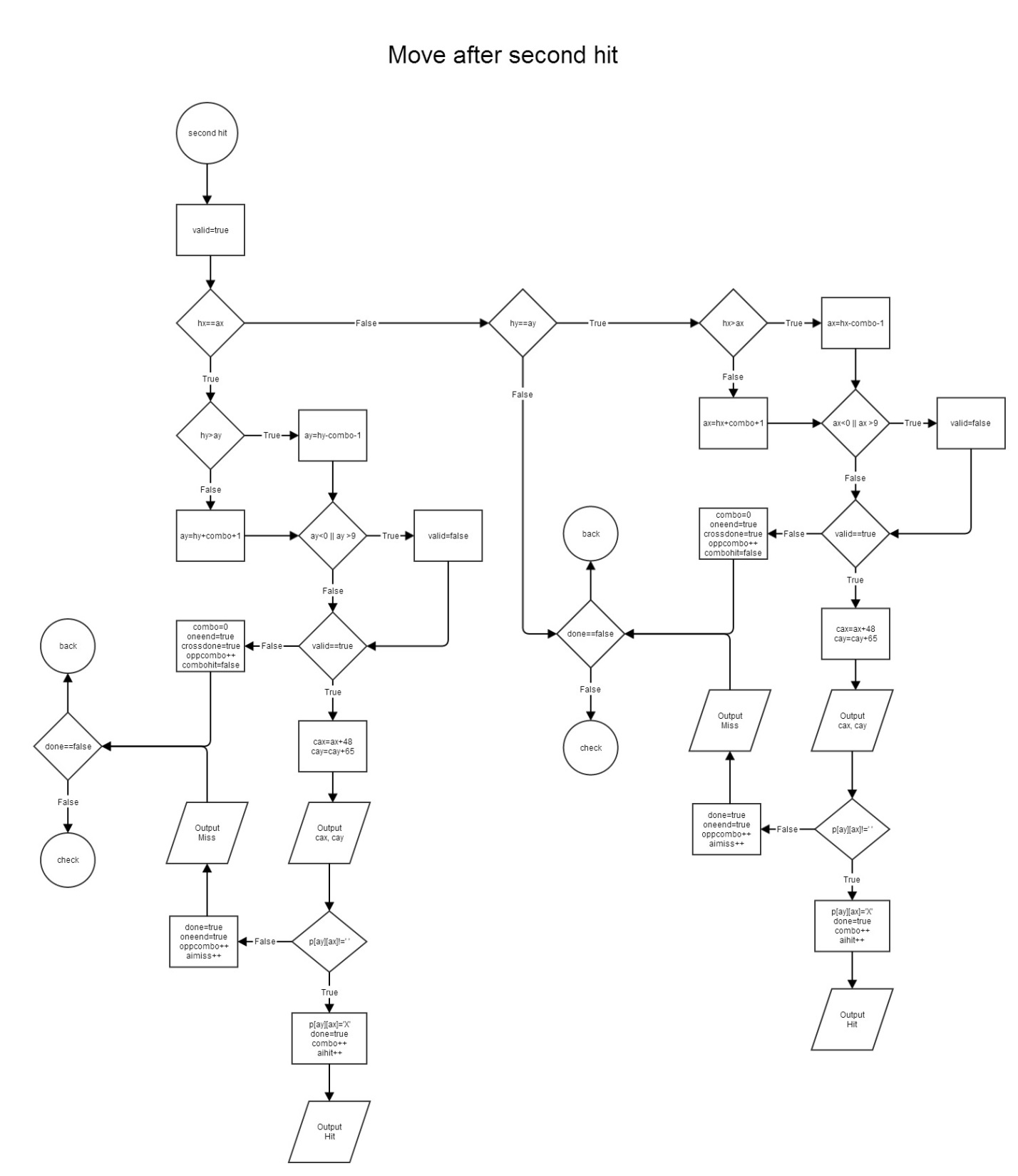
if(hit)

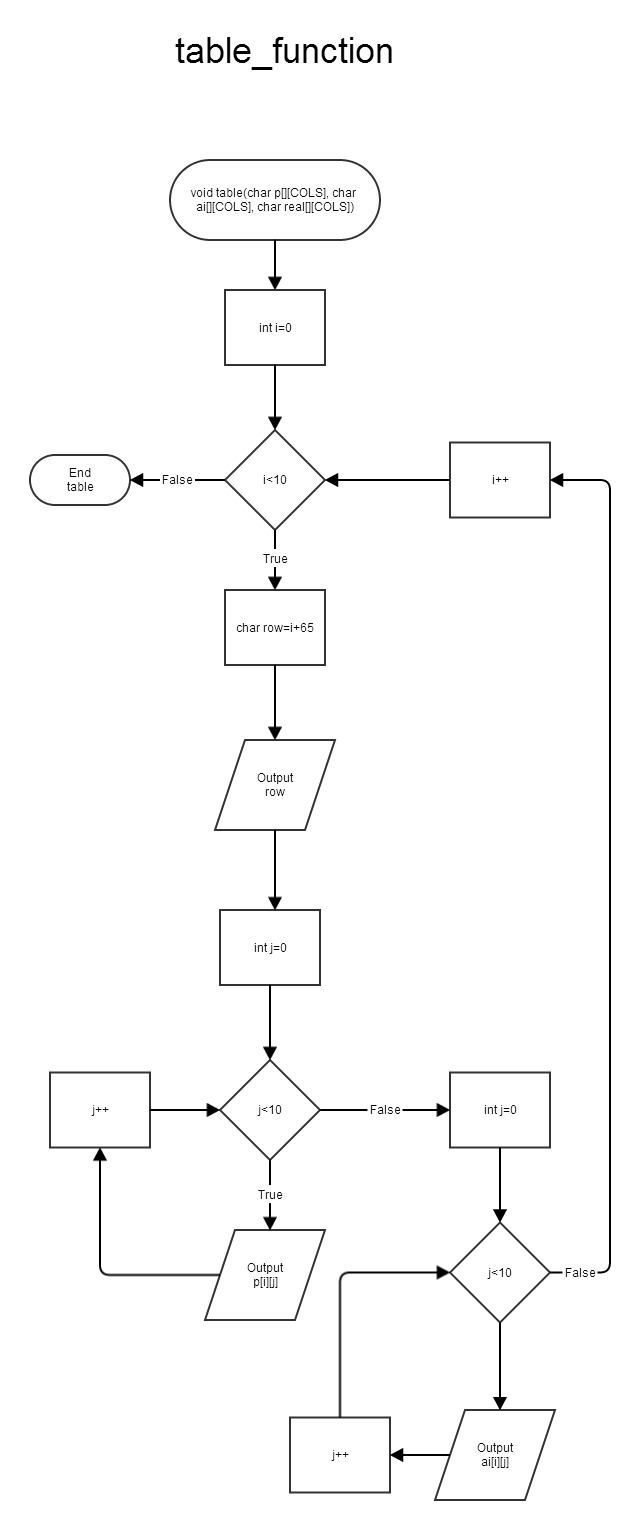
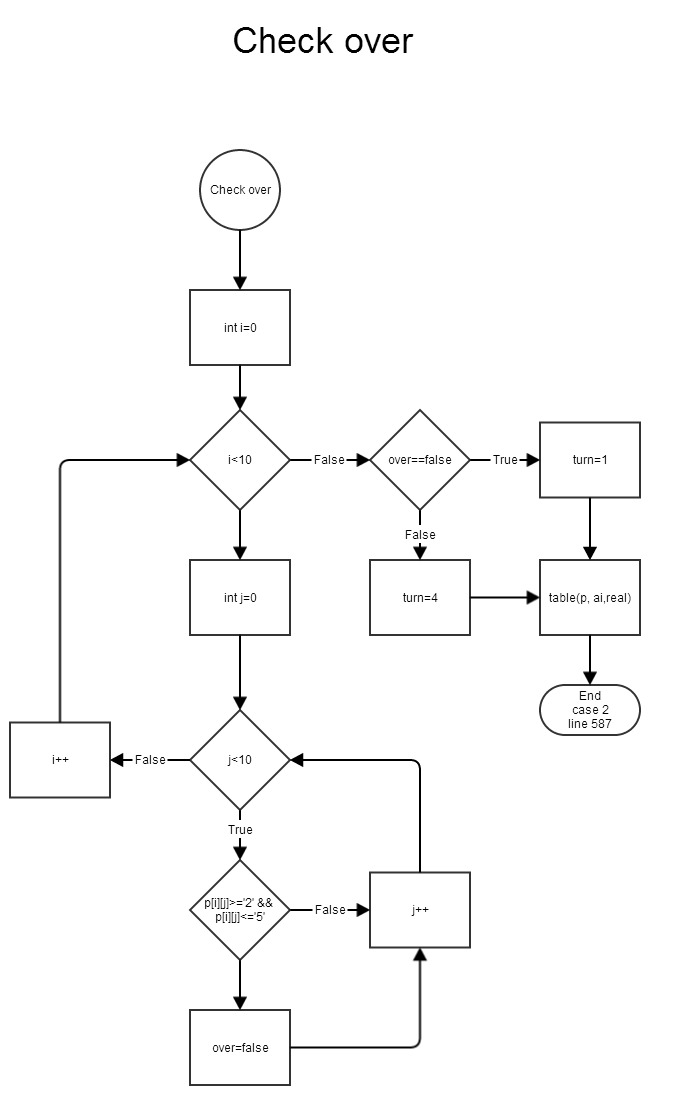
oppcombo+1

}while (not fire)

7. Flowchart







8. Code –

------------- main-------------

\* Table x

\* place x

\* fire x

\* check method x

\* exception for validation c

\* abstract class c

\* template c

\* Problem: Destructor

\* Question: unable to resolve template based identifier

\* solution: create one more class which has class Game

\* missing output binary

\*/

//System Libraries

#include <cstdlib>

#include <iostream>

#include <iomanip>

#include <cctype>

#include <ctime>

#include <fstream>

using namespace std;

//User Libraries

#include "Game.h"

#include "AI.h"

#include "Table.h"

#include "Why.h"

//Global constant

//function prototypes

void intro();

void numTest(int); //exception

//System Begins Here

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

intro();

int num;

bool invalid;

do{

num=0;

invalid=false;

//input size

cout<<"Please enter the size of the table (8-10):";

cin>>num;

try{

numTest(num);

// cout<<"test\n";

}

catch(string numException){

invalid=true;

cout<<numException;

}

}while(invalid==true);

why<test> a(num);

//exit stage right

return 0;

}

//table size validation (exception)

void numTest(int num){

if(num<8 || num>10){

string numException="ERROR: Number less than 7 or greater than 10\n";

throw numException;

}

}

void intro(){

fstream io;

io.open("intro.txt", ios::in);

if(io.is\_open()){

string temp;

while(getline(io, temp))

cout<<temp<<endl;

io.close();

}

}

------------- AI.h -------------

#ifndef AI\_H

#define AI\_H

class AI {

public:

AI();

void delay();

protected:

bool done;

bool cross[4];

bool crossdone;

bool goback;

int hx, hy, x, y;

bool finish;

int oppcombo;

int combo;

char cx, cy;

bool oneend;

bool combohit;

bool bhit;

};

#endif /\* AI\_H \*/

------------- AI.cpp -------------

#include "AI.h"

#include <ctime>

using namespace std;

AI::AI(){

crossdone=true;

finish=true;

goback=true;

done=false;

oppcombo=0;

combo=0;

bhit=true;

for(int i=0;i<4;i++)

cross[i]=true;

}

void AI::delay(){

time\_t tstart, tend;

tstart=time(0);

do{

tend=time(0);

}while(difftime(tend, tstart)<1);

}

------------- Table.h -------------

#ifndef TABLE\_H

#define TABLE\_H

#include <iostream>

using namespace std;

struct table{

char \*\*fake;

char \*\*real;

};

class Table {

public:

//Constructor

Table(int);

int getNum() const{

return num;

}

//return one element of table

char getTable(int x, int y){

return t->real[x][y];

}

void setTable(int x, int y, char i){

t->fake[x][y]=i;

t->real[x][y]=i;

}

//Destructor

virtual ~Table();

protected:

int num;

table \*t;

};

#endif /\* TABLE\_H \*/

------------- Table.cpp -------------

#include "Table.h"

#include <iostream>

using namespace std;

Table::Table(int num){

this->num=num;

t = new table;

t->fake = new char \*[num];

t->real = new char \*[num];

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

t->fake[i] = new char[num];

t->real[i] = new char[num];

}

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

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

t->fake[i][j]=' ';

t->real[i][j]=' ';

}

}

}

Table::~Table(){

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

delete[] t->fake[i];

delete[] t->real[i];

}

delete[] t->fake;

delete[] t->real;

delete t;

}

------------- Game.h -------------

//add abstract class game change functions to virtual

#include <iostream>

#include "Table.h"

#include "AI.h"

using namespace std;

#ifndef GAME\_H

#define GAME\_H

class abstractGame{

public:

virtual void check()=0;

};

class Game:public Table, public AI, public abstractGame{

public:

//Player object

Game(int num);

//ai object with place

Game(int num, char c);

//print table

void print(Game &) const;

//player place

void Place(Game &);

//game start

void start(Game &);

//overloading iostream for debug

friend ostream &operator << (ostream &, const Game &);

~Game();

private:

void placeCheck();

void tableCheck(int);

//Player Fire

void fire(Game &);

//ai fire

void fire(Game &, int);

//check player table

void check();

//check ai table

void check(Game &);

int x1, x2, y1, y2;

int unit[5];

bool over;

bool valid;

float hit, miss;

bool win;

string place;

};

#endif /\* GAME\_H \*/

------------- Game.cpp -------------

#include <fstream>

#include <cstdlib>

#include <ctime>

#include <iostream>

#include <iomanip>

using namespace std;

#include "Game.h"

#include "Table.h"

#include "AI.h"

//overload ostream operator <<

ostream &operator << (ostream &strm, const Game &obj){

strm<<" REAL "<<setw(obj.num\*4+4)<<"FAKE\n";

for(int i=0;i<obj.num;i++){

strm<<setw(4)<<i;

}

strm<<" ";

for(int i=0;i<obj.num;i++){

strm<<setw(4)<<i;

}

strm<<endl;

//third line

strm<<" ";

for(int i=0;i<obj.num;i++){

strm<<"\_\_\_\_";

}

strm<<" ";

for(int i=0;i<obj.num;i++){

strm<<"\_\_\_\_";

}

strm<<endl;

//forth to num line

for(int i=0;i<obj.num;i++){

char row=i+65;

strm<<row<<"| ";

for(int j=0;j<obj.num;j++){

strm<<obj.t->real[i][j];

strm<<" "<<"| ";

}

strm<<" "<<row<<"| ";

for(int k=0;k<obj.num;k++){

strm<<obj.t->fake[i][k];

strm<<" "<<"| ";

}

strm<<endl;

strm<<" ";

for(int l=0;l<obj.num;l++){

strm<<"----";

}

strm<<" ";

for(int m=0;m<obj.num;m++){

strm<<"----";

}

strm<<endl;

}

}

Game::Game(int num):Table(num) {

fstream io;

char temp;

io.open("unit.txt", ios::in | ios::binary);

if(io.is\_open()){

for(int i=0;i<5;i++){

io.read((&temp), sizeof(temp));

unit[i]=temp-48;

}

io.close();

}

hit=0;

miss=0;

win=false;

}

//AI Game Constructor with placed ship

Game::Game(int num, char c):Table(num), AI(){

cout<<"ai c\n";

fstream io;

char temp;

io.open("unit.txt", ios::in | ios::binary);

if(io.is\_open()){

for(int i=0;i<5;i++){

io.read((&temp), sizeof(temp));

unit[i]=temp-48;

}

io.close();

}

//ai place ship

srand(static\_cast<unsigned int>(time(0)));

int count, pos;

bool valid;

for(int q=0;q<5;q++){

do{

valid=true;

count=0;

//random coordinates

y1=rand()%(num-unit[q]); //won't over size

x1=rand()%(num-unit[q]);

pos=rand()%2;

if(pos==0){ //0 horizontal

for(int k=y1;k<y1+unit[q];k++){

if(\*(\*(t->real+k)+x1)==' '){

count++;

}

}

if(count!=unit[q]){

valid=false;

}

if(valid==true){

for(int k=y1;k<y1+unit[q];k++){

\*(\*(t->real+k)+x1)=unit[q]+48;

}

}

}

else{ //1 vertical

for(int k=x1;k<x1+unit[q];k++){

if(t->real[y1][k]==' '){

count++;

}

}

if(count!=unit[q]){

valid=false;

}

if(valid==true){

for(int k=x1;k<x1+unit[q];k++){

t->real[y1][k]=unit[q]+48;

}

}

}

}while(valid==false);

}

// hit=0;

// miss=0;

}

void Game::print(Game &o) const{

cout<<" PLAYER 1"<<setw(num\*4+4)<<"A.I.\n";

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

cout<<setw(4)<<i;

}

cout<<" ";

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

cout<<setw(4)<<i;

}

cout<<endl;

//third line

cout<<" ";

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

cout<<"\_\_\_\_";

}

cout<<" ";

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

cout<<"\_\_\_\_";

}

cout<<endl;

//forth to num line

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

char row=i+65;

cout<<row<<"| ";

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

cout<<t->real[i][j];

cout<<" "<<"| ";

}

cout<<" "<<row<<"| ";

for(int k=0;k<num;k++){

cout<<o.t->real[i][k];

cout<<" "<<"| ";

}

cout<<endl;

cout<<" ";

for(int l=0;l<num;l++){

cout<<"----";

}

cout<<" ";

for(int m=0;m<num;m++){

cout<<"----";

}

cout<<endl;

}

}

void Game::Place(Game &o){

//print table

print(o);

//place ship

for(int q=0;q<5;q++){

do{

do{

do{

valid=true; //reset

cout<<"Choose the coordinates to place the ";

cout<<unit[q]<<"-unit ship with A1A5 form : ";

cin>>place;

try{

placeCheck();

}

catch(string temp){//place ship exception

valid=false;

cout<<temp;

}

}while(valid==false);

cout<<place[0]-65<<place[1]-48<<place[2]-65<<place[3]-48<<endl;

y1=place[0]-65;

y2=place[2]-65;

x1=place[1]-48;

x2=place[3]-48;

cout<<y1<<x1<<y2<<x2<<endl;

try{

tableCheck(q);

}

catch(string temp){

cout<<temp;

}

}while(valid==false);

}while(valid==false);

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

//table

print(o);

}

}

void Game::placeCheck(){

string temp;

if(place.size()!=4){

temp="ERROR: String size\n";

throw temp;

}

else if(place[0]<'A' || place[0]>'J' ||

place[2]<'A' || place[2]>'J' ||

isdigit(place[1]==0 || isdigit(place[3])==0)){

temp="ERROR: Invalid format\n";

throw temp;

}

}

void Game::tableCheck(int q){

int max,min;

int count=0;

string temp;

if(y1==y2){ //x is same

if(abs(x1-x2)!=unit[q]-1){ //check unit invalid

valid=false;

temp="ERROR: Not match units\n";

throw temp;

}

else{ //valid

if(x1>x2){ //check which larger

max=x1;

min=x2;

}

else{

max=x2;

min=x1;

}

// cout<<"max="<<max<<endl;

// cout<<"min="<<min<<endl;

// cout<<"p"<<y1<<endl;

for(int k=min;k<=max;k++){ //check overlap

if(t->real[y1][k]==' '){

count++;

}

}

if(count!=unit[q]){

valid=false;

temp="ERROR: overlap\n";

throw temp;

}

if(valid==true){

for(int k=min;k<=max;k++){

\*(\*(t->real+y1)+k)=unit[q]+48;

}

}

}

}

if(x1==x2){ //y is same

if(abs(y1-y2)!=unit[q]-1){ //check unit

valid=false;

temp="ERROR: not match unit\n";

throw temp;

}

else{ //valid

if(y1>y2){

max=y1;

min=y2;

}

else{

max=y2;

min=y1;

}

// cout<<"max="<<max<<endl;

// cout<<"min="<<min<<endl;

// cout<<"p"<<y1<<endl;

for(int k=min;k<=max;k++){

if(\*(\*(t->real+k)+x1)==' '){

count++;

}

}

if(count!=unit[q]){

valid=false;

temp="ERROR: overlap\n";

throw temp;

}

if(valid==true){

for(int k=min;k<=max;k++){

\*(\*(t->real+k)+x1)=unit[q]+48;

}

}

}

}

if(x1!=x2 && y1!=y2){

valid=false;

temp="ERROR: not horizontal or vertical\n";

throw temp;

}

}

void Game::start(Game &o){

fstream io;

do{

// win=false;

//player fire

fire(o);

if(over==true){

win=true;

cout<<"You win!\n";

io.open("result.txt", ios::out | ios::binary);

char text[]={'R', 'e', 's', 'u', 'l', 't', ' ', 'w', 'i', 'n', '\n'};

io.write(text, sizeof(text));

}

if(win==false){

cout<<"AI fire ";

//ai fire

fire(o, 0);

if(over==true){

win=false;

cout<<"You lose...\n";

io.open("result.txt", ios::out | ios::binary);

char text[]={'R', 'e', 's', 'u', 'l', 't', ' ', 'l', 'o', 's', 'e', '\n'};

io.write(text, sizeof(text));

}

}

}while(over==false);

cout<<"Result outputs to result.txt\n";

}

void Game::fire(Game &o){

string fire;

do{

valid=true;

cout<<"Your turn, enter a coordinate to fire in A0 form :";

cin>>fire;

if(fire.length()!=2){

valid=false;

cout<<"size\n";

}

//fire[1]<'0' || fire[1]]>'9'

else if(fire[0]<'A' || fire[0]>'J' ||fire[1]<'0' || fire[1]>num+48-1){

valid=false;

cout<<"x/y\n";

}

else if(o.t->real[y1][x1]=='O' && o.t->real[y1][x1]=='X'){

valid=false;

cout<<"Overlap\n";

}

}while(valid==false);

y1=fire[0]-65;

x1=fire[1]-48;

//hit

if(o.t->real[y1][x1]>='2' && o.t->real[y1][x1]<='5'){

cout<<"Hit!\n";

o.t->real[y1][x1]='X';

o.t->fake[y1][x1]='X';

hit++;

}

else{

cout<<"Miss...\n";

o.t->real[y1][x1]='O';

o.t->fake[y1][x1]='O';

miss++;

}

//print table

print(o);

//check ai table

check(o);

}

void Game::fire(Game &o, int dum){

srand(static\_cast<unsigned int>(time(0)));

for(int i=0;i<3;i++){

o.delay();

}

int hplan;

o.done=false;

for(int i=0;i<4;i++){

o.cross[i]=true;

}

do{

// cout<<o.crossdone<<" "<<o.finish<<" "<<o.goback<<" "<<o.done<<" "<<o.oppcombo<<" "<<o.combo<<endl;

// cin>>hplan;

// cout<<"o.done = false\n";

o.done=false;

if(o.crossdone==true && o.finish==true &&

o.goback==true && o.done==false &&

o.oppcombo==0 && o.combo==0){

//randon fire

cout<<"random fire\n";

o.goback=true;

do{

cout<<"inner loop(random fire)\n";

valid=true;

// cout<<num<<endl;

o.x=rand()%num;

// cout<<"first rand\n";

o.y=rand()%num;

// cout<<"second rand\n";

// cout<<o.y<<" "<<o.x<<endl;

if(t->real[o.y][o.x]=='O' || t->real[o.y][o.x]=='X'){

valid=false;

cout<<"overlap\n";

}

}while(valid==false);

o.cx=o.x+48;

o.cy=o.y+65;

// cout<<"ai fire";

cout<<o.cy<<o.cx<<"\n";

if(t->real[o.y][o.x]!=' '){

t->real[o.y][o.x]='X';

cout<<"Hit!\n";

o.hx=o.x;

o.hy=o.y;

o.bhit=true;

o.crossdone=false;

o.finish=false;

o.combo=false;

o.done=true;

o.oneend=false;

o.hit++;

}

else{

t->real[o.y][o.x]='O';

cout<<"Miss...\n";

o.done=true;

o.miss++;

}

}

//move after hit

if(o.bhit==true && o.finish==false &&

o.crossdone==false && o.combo==0 &&

o.oppcombo==0 && o.done==false){

do{

cout<<"random cross\n";

o.y=o.hy;

o.x=o.hx;

//check cross rand

hplan=rand()%4;

if(hplan==0) o.y=o.hy-1;

if(hplan==1) o.y=o.hy+1;

if(hplan==2) o.x=o.hx-1;

if(hplan==3) o.x=o.hx+1;

cout<<"hplan = "<<hplan<<endl;

//check over size

if(o.y<0 || o.y>num-1 || o.x<0 || o.x>num-1){

cout<<"Out table\n";

o.cross[hplan]=false;

}

else if(t->real[o.y][o.x]=='X' || t->real[o.y][o.x]=='O'){

cout<<"overlap\n";

o.cross[hplan]=false;

}

if(o.cross[0]==o.cross[1] && o.cross[1]==o.cross[2] &&

o.cross[2]==o.cross[3] && o.cross[0]==false){

cout<<"test all 4 but invalid\n";

o.crossdone=true;

o.finish=true;

o.goback=true;

}

}while(o.crossdone==false && o.cross[hplan]==false);

//valid

if(o.crossdone==false){

cout<<"check hit or miss by cross rand xy\n";

o.cx=o.x+48;

o.cy=o.y+65;

cout<<"ai fire ";

cout<<o.cy<<o.cx<<"\n";

if(t->real[o.y][o.x]!=' '){

t->real[o.y][o.x]='X';

cout<<"Hit!\n";

o.done=true;

o.combo++;

o.crossdone=true;

o.hit++;

}

else{

t->real[o.y][o.x]='O';

cout<<"Miss...\n";

o.done=true;

o.miss++;

}

}

else{

cout<<"crossdone=true, go back to rand\n";

o.goback=true;

}

}

else if(o.combo>0 && o.oneend==false &&

o.done==false && o.crossdone==true){

cout<<"Second hit\n";

valid=true;

if(o.hx==o.x){

cout<<"same x\n";

if(o.hy>o.y) o.y=o.hy-o.combo-1;

else o.y=o.hy+o.combo+1;

if(o.y<0 || o.y >num-1){

valid=false;

}

if(valid==true){

if(t->real[o.y][o.x]=='X' || t->real[o.y][o.x]=='O'){

valid=false;

}

if(t->real[o.y][o.x]=='O'){

o.finish=true;

o.goback=true;

o.crossdone=true;

o.combo=0;

o.oppcombo++;

o.oneend=true;

}

if(valid==true){

o.cx=o.x+48;

o.cy=o.y+65;

// cout<<"ai fire ";

cout<<o.cy<<o.cx<<"\n";

if(t->real[o.y][o.x]!=' '){

t->real[o.y][o.x]='X';

cout<<"Hit!!!\n";

o.done=true;

o.combo++;

o.hit++;

}

else{

t->real[o.y][o.x]='O';

cout<<"Miss...\n";

o.done=true;

o.oneend=true;

o.oppcombo++;

o.miss++;

}

}

}

else{ //check ->GO TO OPPCOMBO

cout<<"next xy invalid change to opposite side\n";

o.combo=0;

o.oneend=true;

o.crossdone=true;

o.oppcombo++;

o.combohit=false;

}

}

if(o.hy==o.y){

cout<<"same y\n";

if(o.hx>o.x) o.x=o.hx-o.combo-1;

else o.x=o.hx+o.combo+1;

if(o.x<0 || o.x >num-1){

valid=false;

o.combo=0;

o.goback=true;

o.finish=true;

}

if(valid==true){

if(t->real[o.y][o.x]=='X' || t->real[o.y][o.x]=='O'){

valid=false;

o.finish=true;

o.goback=true;

}

if(valid==true){

o.cx=o.x+48;

o.cy=o.y+65;

// cout<<"ai fire ";

cout<<o.cy<<o.cx<<"\n";

if(t->real[o.y][o.x]!=' '){

t->real[o.y][o.x]=='X';

o.combo++;

o.done=true;

o.hit++;

}

else{

t->real[o.y][o.x]=='O';

cout<<"Miss...\n";

o.done=true;

o.oneend=true;

o.oppcombo++;

o.combo=0;

o.combohit=false;

cout<<"oneend==true\n";

cout<<"done==true\n";

o.miss++;

}

}

}

if(valid==false){ //GO TO OPPCOMBO

cout<<"next xy invalid change to other side\n";

o.combo=0;

o.oneend=true;

o.crossdone=true;

o.oppcombo++;

o.combohit=false;

}

}

}

//check other side

else if(o.oppcombo>0 && o.oneend==true && o.done==false){

cout<<"one side end check other side\n";

cout<<"oppcombo = "<<o.oppcombo<<endl;

if(o.hx==o.x){

cout<<"same x\n";

if(o.combohit==false){

if(o.hy>o.y) o.y=o.hy+o.oppcombo;

else o.y=o.hy-o.oppcombo;

}

else{

if(o.y>o.hy) o.y=o.hy+o.oppcombo;

else o.y=o.hy-o.oppcombo;

}

cout<<o.y<<o.x<<endl;

if(o.y<0 || o.y>num-1 || t->real[o.y][o.x]=='O' ||

t->real[o.y][o.x]=='X'){

o.oppcombo=0;

o.goback=true;

o.finish=true;

o.crossdone=true;

o.combo=0;

o.done=false;

cout<<"overlap or oversize\n";

}

else{

o.cx=o.x+48;

o.cy=o.y+65;

// cout<<"ai fire ";

cout<<o.cy<<o.cx<<"\n";

if(t->real[o.y][o.x]!=' '){

t->real[o.y][o.x]='X';

cout<<"Hit!!!\n";

o.done=true;

o.oppcombo+=1;

o.combohit=true;

o.hit++;

}

else{

t->real[o.y][o.x]='O';

cout<<"Miss...\n";

o.done=true;

o.combo=0;

o.oppcombo=0;

o.finish=true;

o.goback=true;

o.crossdone=true;

o.miss++;

}

}

}

else if(o.hy==o.y){

cout<<"same y\n";

if(o.combohit==false){

if(o.hx>o.x) o.x=o.hx+o.oppcombo;

else o.x=o.hx-o.oppcombo;

}

else{

if(o.x>o.hx) o.x=o.hx+o.oppcombo;

else o.x=o.hx-o.oppcombo;

}

cout<<o.y<<o.x<<endl;

if(o.x<0 || o.x>num-1 || t->real[o.y][o.x]=='O' ||

t->real[o.y][o.x]=='X'){

o.oppcombo=0;

o.goback=true;

o.finish=true;

o.crossdone=true;

o.combo=0;

o.done=false;

cout<<"overlap or oversize\n";

}

else{

o.cx=o.x+48;

o.cy=o.y+65;

// cout<<"ai fire ";

cout<<o.cy<<o.cx<<"\n";

if(t->real[o.y][o.x]!=' '){

t->real[o.y][o.x]='X';

cout<<"Hit!!!\n";

o.done=true;

o.oppcombo+=1;

o.combohit=true;

o.hit++;

}

else{

t->real[o.y][o.x]='O';

cout<<"Miss...\n";

o.done=true;

o.combo=0;

o.oppcombo=0;

o.finish=true;

o.goback=true;

o.crossdone=true;

o.miss++;

}

}

}

}

}while(o.done==false);

//print table

print(o);

//check player table

check();

}

//check player table

void Game::check(){

over=true;

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

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

if(t->real[i][j]>='2' && t->real[i][j]<='5')

over=false;

}

}

}

//check ai table

void Game::check(Game &o){

over=true;

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

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

if(o.t->real[i][j]>='2' && o.t->real[i][j]<='5')

over=false;

}

}

}

Game::~Game(){

// for (int i=0;i<num;i++){

// delete[] t->fake[i];

// delete[] t->real[i];

// }

// delete[] t->fake;

// delete[] t->real;

// delete t;

}

------------- Why.h -------------

#ifndef WHY\_H

#define WHY\_H

#include "Game.h"

#include "AI.h"

#include "Table.h"

#include <iostream>

using namespace std;

template <class T>

class why{

public:

why(int num){

this->num=num;

t = new T(num);

}

int num;

~why(){

delete t;

}

private:

T \*t;

};

class test{

public:

test(int num){

Game p(num);

Game ai(num, 'c');

p.Place(ai);

p.start(ai);

}

~test(){}

};

#endif /\* WHY\_H \*/