BÁO CÁO ĐỒ ÁN CUỐI KÌ: GAME CARO

Họ và Tên: Nguyễn Hữu Lương Anh

MSSV: 1712277

Lớp: 17CTT2

I. Các kỹ thuật trong đồ án

1. Kỹ thuật di chuyển con trỏ lên xuống trong menu
2. Kỹ thuật chọn menu
3. Kỹ thuật kiểm tra thắng thua
4. Kỹ thuật đếm các thông số của trò chơi
5. Kỹ thuật load và save file
6. Kỹ thuật chơi với máy
7. Kỹ thuật chạy chữ qua lại khi thắng

II. Mô tả các đoạn mã và các ý của từng chức năng trong đồ án

1. Menu

- Menu của trò chơi được viết tại hàm main bao gồm các hàm:

- `g.nameGameXO()`: In ra màn hình giao diện cơ bản của Menu: Tên game, X,O.
- `g.displayMenu()`: In ra màn hình các lựa chọn trong menu và phần How To Play
- `while (cur){…}`: Vòng lặp while này là đề thực hiện các thao tác di chuyển và chọn Menu. Được tham khảo từ vòng lặp while trong phần gợi ý từ file pdf hướng dẫn làm đồ án. Gồm các phần cơ bản như hàm `g.waitKeyBoard()` để nhận giá trị từ bàn phím, `g.getCommand()` để kiểm tra giá trị từ bàn phím.
  - ➔ Khi người dùng nhập S hoặc W, kiểm tra biến check (biến kiểm tra xem con trỏ đang ở vị trí nào của menu), nếu check < 4 hoặc check > 1 (mục đích để con trỏ không đi quá menu) thì thực hiện hàm `g.moveMenuDown()` và `g.setPointer()` để cập nhật lại tọa độ _x, _y và dịch chuyển con trỏ theo yêu cầu. Sau đó tăng, giảm biến check và thực hiện lần lượt các hàm `g.deleteColorMenu(check)` và `g.setColorMenu(check)` để xóa màu chữ của menu trước khi di chuyển xuống, lên và cập nhật là màu chữ menu hiện đang chọn.
  - ➔ Khi người dùng nhấn Enter, tức `case 13` thì chỉnh lại `cur = false` để vòng lặp sau khi thực hiện các hàm bên dưới sẽ kết thúc Menu. Tại đây, sẽ kiểm tra các biến check, nếu check bằng vị trí tương ứng nào của Menu thì sẽ thực hiện thao tác đó.
  - ➔ Nếu người dùng chọn Play game hoặc Load game thì sẽ thực hiện hàm `g.settingGame()` đầu tiên với mục đích hiện ra một Menu nhỏ để chọn giữa chơi với máy hoặc với người. Cách thực hiện cũng tương tự như thực hiện với Menu lớn ban đầu và nếu người dùng chọn cách chơi nào thì sẽ lưu vào biến `bool _checkPlayWithMan` . Sau đó thì thực hiện `g.startGame(), g.displayEffect()` để bắt đầu trò chơi và hiện thị giao diện. Tiếp ta kiểm tra biến `bool _checkPlayWithMan`, rồi cho chạy `g.playGame()` và `g.playGameWithCom()` tương ứng.

➔ Nếu người dùng chọn About, thực hiện `system("cls")` và `g.displayAbout()` để xóa màn hình Menu hiện tại và in ra màn hình phần About. Sau đó chờ người dùng nhập vào bàn phím, nếu chọn phím B thì sẽ quay lại Menu ban đầu bằng cách set lại cái thông số như ban đầu (`cur = true`, …).

➔ Nếu chọn Exit thì sẽ thoát trò chơi

2. *Chơi game*

- `g.playGame()`: Chơi 2 người

• `g.playGame()` về cơ bản tương tự như trong file hướng dẫn, thêm vào 2 trường hợp nhập L và T từ bàn phím để Save Game và Load Game

- `g.playGameWithCom()`: Chơi với máy

• Trong `if (processCheckBoard()){..}` ta thay đổi thêm `case 2` tại `processFinish()`. Thêm hàm `setCom()` để máy đánh và thực hiện lại các hàm kiểm tra thắng thua hòa như ở phần trên . Trong `setCom()` có gọi thêm 1 hàm `_b->setBoardCom(_x, _y)` tử class _Board.

• `_b->setBoardCom(_x, _y)` :

➔ Chuyển từ tọa độ console sang tọa độ bàn cờ. Sau đó lần lượt kiểm tra và đánh theo dòng, cột và 2 đường chéo.

```
int Col = (pX - 2) / 4;
int Row = (pY - 1) / 2;
```

➔ Vì các phần tương tự nhau nên em chỉ trình bày kiểm tra và đánh theo dòng

➔ Vòng for đầu tiên chạy từ vị trí đánh sang phải, đến khi nào gặp được điểm check = - 1 (tức X) thì tăng biến đếm check lên. Nếu không, thì kiểm tra tại vị trí không phải check = -1 đó bằng 0 hay không, nếu có gán biến temp = j, không thỉ break vòng for.

```
int check = 0;

int temp = 0;

for (int j = Col; j < _size; j++){
	if (_pArr[Row][j].getCheck() == -1){
		check++;
	}
	else {
		if (_pArr[Row][j].getCheck() == 0)
			temp = j;
		break;
	}
}
```

➔ Vòng for thứ 2 chạy từ vị trí đánh – 1 sang bên phải, và cũng thực hiện như vòng lặp đầu tiên.

```
for (int j = Col - 1; j >= 0; j--){
	if (_pArr[Row][j].getCheck() == -1){
		check++;
	}
```

```
        else {
                if (_pArr[Row][j].getCheck() == 0)
                        temp = j;
                break;
        }
}
```

➔ Sau đó, kiểm tra xem biến check có lớn hơn bằng 3 không, mục đích để chặn đầu khi người dùng đã đánh 3 lần X kề nhau.

```
if (check >= 3){
        _pArr[Row][temp].setCheck(1);
        _Common::gotoXY(4 * temp + 2, 2 * Row + 1);
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 0x3);
        cout << "O";
        return 1;
}
```

➔ Các phần kiểm tra theo cột và 2 đường chéo thực hiện tương tự

➔ Trường hợp mà người dùng đánh nhỏ hơn 3 nước kề nhau. Vòng lặp đầu tiên vẫn chạy từ vị trí đánh sang phải. Nếu vị trí hiện tại check = -1 thì checkRowX tăng lên. Ngược lại, kiểm tra tiếp tại vị trí đó đã đánh hay chưa, nếu chưa thì gán temp = j, còn đã đánh rồi thì chạy vòng for tiếp tục, kiểm tra O, nếu O liên tục lớn hơn bằng 2 mà vị trí kế tiếp chưa đánh thì đánh tại vị trí đó (tấn công đơn giản). Tương tự cho for từ vị đánh – 1 sang trái.

```
int checkRowX = 0;
int checkRowO = 0;
int temp = 0;
int temp1 = 0;
for (int j = Col; j < _size; j++){
        if (_pArr[Row][j].getCheck() == -1){
                checkRowX++;
        }
        else {
                if (_pArr[Row][j].getCheck() == 0){
                        temp = j;
                        break;
                }
                for (int m = j; m < _size; m++){
                        if (_pArr[Row][m].getCheck() == 1){
                                checkRowO++;
                        }
                        else
                        {
                                if (_pArr[Row][j].getCheck() == 0){
                                temp1 = m;
                                }
                        break;
                        }
                }
        break;
        }
}
```

```
if (checkRowX < 3){
	if (checkRowO >= 2){
	_pArr[Row][temp1].setCheck(1);
	_Common::gotoXY(4 * temp1 + 2, 2 * Row + 1);
	SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 0x3);
	cout << "O";
	return 1;
}
	_pArr[Row][temp].setCheck(1);
_Common::gotoXY(4 * temp + 2, 2 * Row + 1);
SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 0x3);
cout << "O";

return 1;
}
```

3. *Load Game và Save Game*

- **Load Game**:

- Đầu tiên thực hiện các câu lệnh đọc file.
- Thực hiện câu lệnh `startGame()` để bắt đầu trò chơi. Thực hiện 2 vòng for chạy hết bàn cờ, lần lượt đọc vào biến check. Rồi set Check tại vị trí đó, và in X và O tương ứng check lên bàn cờ. Cho một biến sum tính tổng các lượt check. Nếu sum == -1 thì lượt đi trước khi save game là X, vậy lượt đánh tiếp theo là O(turn = false).

```
startGame();
for (int i = 0; i < _b->getSize(); i++){
	for (int j = 0; j < _b->getSize(); j++){
		inFile >> check;
		_b->setCheckBoard(i, j, check);
		_b->loadBoard(i, j);
		sum += check;
	}
}

if (sum == -1){
_turn = false;
}
```

- Rồi lần lượt đọc các thông số như số lượt đi 2 bên, số trận thắng thua.

```
int count;
inFile >> count;
_count1 = count;
inFile >> count;
_count2 = count;
inFile >> count;
_countWin1 = count;
inFile >> count;
_countWin2 = count;
```

- **Save Game:**

- Thực hiện 2 vòng for để ghi từng check của từng vị trí vào file. Ghi các thông số cũng tương tự như vậy.

```
for (int i = 0; i < _b->getSize(); i++){
	for (int j = 0; j < _b->getSize(); j++)
		outFile << _b->getCheckAt(i, j) << " ";
}

outFile << _count1 << " " << _count2 << " " << _countWin1 <<" "<< _countWin2
<< " " << _turn;
```

4. *Kiểm tra thắng thua*

- `__b->testBoard(_x,_y):`

- Chuyển đổi từ tọa độ console sang tọa độ bàn cờ

```
int Col = (pX - 2) / 4;
int Row = (pY - 1) / 2;
```

- Kiểm tra tại vị trí hiện tại check có bằng 0 hay không, nếu bằng thì return lại 2

```
if (_pArr[Row][Col].getCheck() == 0)
	return 2;

int count = 0;
// Kiem tra de chan 2 dau
int check = 0;
```

- Rồi lần lượt chạy từ vị trí hiện tại sang phải, nếu check khác check hiện tại thì thực hiện kiểm tra tiếp vị trí đang xét có bằng X hay O không, nếu có tăng check lên rồi break vòng lặp(mục đích của check để kiểm tra chặn 2 đầu). Còn nếu check bằng check hiện tại thì tăng biến count(đếm số lần X hay O liên tiếp) lên. Tương tự kiểm tra từ vị trí hiện tại -1 sang trái.

```
for (int j = Col; j < _size; j++){
if (_pArr[Row][j].getCheck() != _pArr[Row][Col].getCheck())
{
	if (_pArr[Row][j].getCheck() == 1 &&_pArr[Row][Col].getCheck() == -1
|| (_pArr[Row][j].getCheck() == -1 && _pArr[Row][Col].getCheck() == 1))
		check++;
break;
}
else
	count++;
}
```

- Kiểm tra count lớn hơn bằng 5 và check không bằng 2 thì return -1 tức X thắng, và ngược lại

```
if (count >= 5 && check != 2) {
	if (_pArr[Row][Col].getCheck() == -1)
		return -1;
	else
		return 1;
}
```

- Kiểm tra cột và 2 hàng chéo tương tự.

- Sau khi kiểm tra thắng thua thì kiểm tra hòa. Chạy 2 for hết bàn cờ tăng biến đếm coutSize lên, nếu vị trí nào bằng 0 thì return 2.

```
int countSize = 0;
for (int i = 0; i < _size; i++){
	for (int j = 0; j < _size; j++){
		countSize++;
		if (_pArr[i][j].getCheck() == 0)
			return 2;
	}
}
```

- Kiểm tra countSize có bằng diện tích của bàn cờ không. Nếu bằng thì trò chơi hòa, return 0.

```
int area = _size*_size;
if (countSize == area)
	return 0;
```

5. *Xử lý hiệu ứng thắng thua*

- `makeEffectWin(pWhoWin)` đặt trong các case của processFinish() với mục đích in ra hiệu ứng thắng thua.
- Sử dụng vòng lặp while(true) để tạo hiệu ứng chữ. Tạo một class mới tên Random để tạo random màu liên tục khi chạy vòng while.
- Với 2 chữ WIN và LOSE, ta chỉ cần chạy vòng while đổi màu chữ và in đè lại.
- Với chữ Player 1, 2 Win có hiệu ứng chạy qua lại

```
if (turn == true){
	_Common::gotoXY(X - 1, Y);
	cout << " ";
}

if (turn == false) {
	_Common::gotoXY(X + n, Y);
	cout << " ";
}

_Common::gotoXY(X, Y);
	cout << str;
```

➔ turn là biến kiểu bool. turn = true chữ đang chạy từ trái sang phải, turn = false chữ đang chạy phải sang trái. Mỗi lần chạy vòng while, thì chữ sẽ được dịch sang theo turn. Nếu turn = true, thì ngay tại vị trí chữ trước khi dịch chuyển phải in khoảng trắng để làm mất chữ, với false thì ngược lại.

```
if (X <= 86){
	turn = true;
}
else if (X >= 99){
	turn = false;
}

if (turn == true){
	X++;
}
```

```
else if (turn == false){
       X--;
}
Sleep(400);

count++;
if (count == 34)
              break;
```

- ➔ Khi tọa độ X vượt qua biên trái (86) thì set lại turn = true để chữ chạy tới vị trí 86 thì quay đầu lại, tương tự biên phải.
- ➔ Kiểm tra lại turn, nếu turn = true thì X sẽ được tăng
- ➔ Sleep(400) để in ra màn hình chậm
- ➔ Biến count dùng để đếm số lần chữ dịch chuyển, khi count = 34 thì hiệu ứng sẽ dừng lại

III. Nguồn tham khảo

Tham khảo chỉnh hiệu ứng chữ từ video:

https://www.youtube.com/watch?v=C7yokRqdd4o&feature=share

Trong video, hướng dẫn chuyển động lên xuống và mỗi lần chuyển động là làm mới lại console (clear screan) , tuy nhiên trong bài em sử dụng trái phải và mỗi lần chuyển động không làm mới lại console(cụ thể có ở trên).