프로젝트 #1

과 목 명 : 고급프로그래밍(가)

학　　번 : 20170358

이　　름 : 김 희 수

지도교수 : 이 길 호

1. 문제 파악

4 * 4 배열로 이루어진 카드 뒤집기 게임을 만들려고 하기 때문에 , 가장 중요한 것은 카드의 정보를 가지고 있는 4*4 배열이다. 그렇다면 4*4 배열을 어떤 식으로 몇개의 배열을 선언할 것인가도 중요한데, 문제에서 고정배치와 랜덤배치 , 카드의 뒷면 그리고 선택한 카드에 대한 정보를 저장하고 있어야 하기 때문에 3가지의 배열을 선언하였다. 카드의 앞면을 가지고 있는 배열, 뒷면을 가지고 있는배열 그리고 카드가 발견된 카드인지 저장하는 bool타입 배열이다.

```
    // 카드내용의 유지와 게임 진행을 위한 카드배열
    char behindmap[4][4];
    char displayMap[4][4];
    bool opend[4][4];
```

그리고 랜덤 배치 기능을 집어 넣으라고 했는데, 수가 2번 이상 중복되면 안되므로 인덱스가 8까지 있는 int형 배열을 하나 선언해 준다.

```
// 랜덤배치시 사용하는 배열
    int check[9];
```

그렇다면 이 배열들을 이용해서 카드를 어떻게 생성해 낼 것인가, 우선 나는 cardgame이라는 class를 만들었다. 그렇다면 이 클래스에 구현해야 할 함수는 무엇이 있을까, 우선 게임을 진행해야 하므로 게임을 시작하는 함수가 필요할 것이다. 그리고 점수를 세는 방식에 있어서 규칙이 필요하니 규칙을 가지고 있는 점수를 세는 함수를 만들 것이다. 그리고 게임이 끝났을 때 문구를 출력하는 함수, 게임 진행중에 카드를 출력해줄 함수 그리고 추가적으로 내가 고른 카드가 이미 짝을 가진 카드인지 확인해주는 함수를 만들 것이다. 마지막으로 저 위의 3개의 배열을 초기화 해 줄 게임 시작과 동시에 이루어지는 생성자를 선언할 것이다.

```
    cardgame(int gamemode); // 생성자
    void gameStart(); // 게임 시작 함수
    bool checkMap(); // 카드 체크 함수
    void printMap(); // 전체 카드 출력 함수
    void countingScore(int Pos1, int Pos2, int& Score, int player); // 점수
카운팅함수
    void gameOver(); // 게임종료 함수
```

그리고 게임을 진행하면서 클래스 내에 업데이트 해줘야하는 사항에 대한 변수를 다음과 같이 선언하여 준다.

```
    int p1Score; // 점수 관련 변수
    int p2Score;
    int firstPos; // 첫번째 입력 (카드의 위치)
    int secondPos; // 두번째 입력
    int turns; // 총 게임의 턴수
    bool gameStatus; // 게임의 현재 상태
    bool checked; // 카드의 짝이 있는지 저장
```

2. 프로그램 설계

우선 생성자에서 배열을 초기화 하는 것부터 진행해야 한다. 코드는 다음과 같이 작성했다.

```
cardgame::cardgame(int gamemode){
    for(int i = 0; i < 9; i++){
        check[i] = 0;
    }

    for(int i = 0; i < 4; i ++){
        for(int j = 0; j < 4; j ++){
            displayMap[i][j] = 'x';
        }
    }

    for(int i = 0; i < 4; i ++){
        for(int j = 0; j < 4; j ++){
            opend[i][j] = false;
        }
    }
    if(gamemode == 1){
        int input = 1;
        for(int i = 0; i < 4; i ++){
            for(int j = 0; j < 4; j ++){
                behindmap[i][j] = '0' + input % 10;
                if( i <= 1){
                    input ++;
                } else{
                    input --;
                }
                if(input > 8) input --;
            }
        }
    }else{
        checked = true;
        for(int i = 0; i < 4; i ++){
            for(int j = 0; j < 4; j ++){
                while(checked){
                    int input = rand()%8 + 1;
                    if(check[input] < 2){
                        check[input]++;
                        behindmap[i][j] = '0' + input % 10;;
                        checked = false;
                    }
                }
                checked = true;
            }
        }
    }
}
```

우선 인자로 game모드라는 인자를 받아 1은 static 모드 2는 동적 모드이다. 우선 카드에 동적으로 숫자를 넣을 때 2번이상 숫자가 들어가지 않게 하기 위한 check 배열 그리고 카드의 숫자와 관계없이 static모드와 dynamic 모드 모두 똑같은 카드의 뒷면인 displayMap 배열 그리고 카드의 짝이 찾아진 상태인지 아닌지를 표시하는 bool 타입의 opend 배열을 초기화 해준다. 이후 인자로 받은 게임모드로 카드의 숫자를 어떻게 초기화 할건지 정해주어야 하는데 우선 char형의 배열이기 때문에 '0'을 기준으로 우리가 넣고자 하는 숫자를 10으로 나눈 나머지만큼 더해서 배열을 초기화 해준다. Dynamic mode 같은 경우에는 랜덤으로 값을 추출해서 집어 넣는 것이기 때문에 숫자를 집어넣을 때마다 check 배열의 집어넣는 숫자의 인덱스의 값을 1씩 증가시켜 2번이상 들어가지 않게 처리해주고 2번 이상일 경우에는 숫자를 다시 뽑아주는 것을 반복하였다.

printMap() 함수는 매우 간단하다. 정해진 틀에 그저 2차원 배열을 2중 중첩 포문으로 해결을 하였다.

```
void cardgame::printMap(){
    cout << "   1234" << endl;
    cout << "-+-----" << endl;
    for(int i = 0; i < 4  ; i ++){
        cout << i+1 << "| ";
        for(int j = 0; j < 4; j++){
            cout << displayMap[i][j];
        }
        cout << endl;
    }
}
```

checkMap() 함수는 카드가 전부 뒤집어진 상태인지 안뒤집어진 상태인지 판단해주는 함수이다. 함수의 내부는 다음과 같다. 카드는 두쌍이 같아야 최종적으로 뒤집어지기 때문에 display Map에 만약 전부 뒤집어진 카드밖에 없다면 게임이 종료된 것으로 판단된다. 또한 turn 이 0번 남았을 때에도 마찬가지이다.

```
bool cardgame::checkMap(){
    if(turns == 0){
        return false;
    }
    for(int i = 0; i < 4; i++){
        for(int j = 0; j < 4; j ++){
            if(displayMap[i][j] == 'x') return true;
        }
    }

    return false;
}
```

countingScore(int, int , int& ,int) 함수는 카드게임에서 점수를 세는 함수이다. 이 함수는 카드 게임에서 가장 많은 논리식을 포함하고 있고, 가장 중요한 함수이다 . 각각 인자는 입력받은 첫번째 위치와, 두번째 위치 그리고 점수, 그리고 플레이어 번호이다. 먼저 이 함수의 코드는 다음과 같다.

```
void cardgame::countingScore(int Pos1, int Pos2, int& Score, int player){
    int firstX;
    int firstY;
    int secX;
    int secY;

    firstX = Pos1 / 10 - 1;
    firstY = Pos1 % 10 - 1;

    secX = Pos2 / 10 - 1;
    secY = Pos2 % 10 - 1;

    bool test = true;
    int tempPos1;
    int tempPos2;
    while(test){
        test = false;

        if(firstX > 3 | firstY > 3 | secX > 3 | secY > 3){
            cout << "Your Pair is out of Range" << endl;
            test = true;
        }

        if(opend[firstX][firstY] == true | opend[secX][secY] == true){
            if(opend[firstX][firstY] == true){
                cout << firstX + 1 << "," << firstY + 1 << " ";
            }
            if(opend[secX][secY] == true){
                cout << secX + 1 << "," << secY + 1 << " ";
            }
            cout << "your pair was already selected" << endl;

            test = true;
        }

        if(test){
            cout << "choose again cards : ";
            cin >> tempPos1 >> tempPos2;

            firstX = tempPos1 / 10 - 1;
            firstY = tempPos1 % 10 - 1;

            secX = tempPos2 / 10 - 1;
            secY = tempPos2 % 10 - 1;

        }
```

```
    }
    displayMap[firstX][firstY] = behindmap[firstX][firstY];
    displayMap[secX][secY] = behindmap[secX][secY];
    printMap();

    if(player == 1){
        cout << "P1";
    } else {
        cout << "P2";
    }
    if(displayMap[firstX][firstY] == displayMap[secX][secY]){
        opend[firstX][firstY] = true;
        opend[secX][secY] = true;
        Score++;
        cout << " found a matching pair! P1's score: " << p1Score << ",
P2's score: "<< p2Score << endl;
        return;
    }

    cout << " failed to find a matching pair" << endl;

    displayMap[firstX][firstY] = 'x';
    displayMap[secX][secY] = 'x';

    return;
}
```

먼저 입력받은 좌표값에 대해서 좌표를 입력받는 방식이 정수로 입력받기 때문에 그것을 분리 해주기 위해서 나머지와 몫을 각각의 변수에 저장한 후에, 반복문으로 넘어간다. 반복문에서는 내가 고른 카드에 대해서 두가지 오류를 검출하는데, 첫번째로는 이 좌표가 범위에서 벗어난 좌표가 아닌지, 그리고 이 좌표가 직전이나 이전에 선택되어 이미 공개된 카드가 아닌지에 대해서 검사하고 만약 두가지 중 한가지라도 해당할 경우 좌표를 재 입력받는 시스템을 차용하였다. 이후 사용자에게 보여지는 배열에 우리가 선택한 좌표의 카드의 숫자를 대입하고 맵을 보여준 후 두개의 값이 같으면 점수를 올려주고 점수를 공개하는 방식의 코드 그리고 만약 두개가 다르다면 다시 원래대로 카드를 뒤집어 놓는 방식을 선택하였다.

그리고 이들을 전부 사용하고 있는 함수가 바로 gameStart()함수이다.

gameStart()함수는 생각보다 간단하다. 다음과 같이 작성하였다.

```
void cardgame::gameStart(){
    gameStatus = true;
    turns = 20;

    printMap();

    while(gameStatus){
```

```
        cout << "P1's turn, choose two cards (" << turns << " turns remain)
:";
        cin >> firstPos >> secondPos;

        countingScore(firstPos, secondPos, p1Score, 1);

        turns --;

        gameStatus = checkMap();
        if(!gameStatus) break;

        cout << "P2's turn, choose two cards (" << turns << " turns remain)
:";
        cin >> firstPos >> secondPos;

        countingScore(firstPos, secondPos, p2Score, 2);

        turns --;

        gameStatus = checkMap();
    }

    gameOver();
}
```

먼저 게임을 시작하였기 때문에 게임의 상태와 턴을 초기상태로 초기화 시켜준다. 이후 while문으로 계속해서 게임을 진행하는데 처음에는 p1의 입력을 두번째에는 p2의 입력을 받는 식으로 반복한다. 그리고 각 플레이어의 턴이 끝날때마다 점수체크, 턴 감소, 그리고 맵 확인을 통한 게임 상태 업데이트를 시켜준다. 그리고 p1과 p2 사이에 If문을 집어넣어 만약 p1이 마지막 카드를 선택했을때 게임이 종료될 수 있도록 하였다. 그리고 게임이 종료되면 사용자들의 최종 점수를 알려주는 gameover() 함수를 호출하며 gameStart()함수는 종료가 된다 그리고 이를 main함수에서 클래스를 선언하여 cin으로 게임모드를 입력받고 게임을 진행할 수 있도록 하였다.

```
int main(void){
    int gamemode;

    cout << "Choose card generation mode (1: static, 2:Random): ";
    cin >> gamemode;

    cardgame game(gamemode);

    game.gameStart();

    return 0;

}
```