프로젝트 2048 보고서 20171637 박성우

함수

1. Void rotate(int b[4][4])

c[4][4]라는 이차배열을 만들어 b[4][4]를 시계방향으로 90도 회전한 이차배열을 c[4][4] 에 대입한다. 시계 방향으로 90도 회전시키는 것의 예는 다음과 같다.

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | 4 | 6 | 8 |
| 8 |  |  |  |
| 4 |  |  |  |
| 0 |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 4 | 8 | 2 |
|  |  |  | 4 |
|  |  |  | 6 |
|  |  |  | 8 |

c[4][4]에 시계방향으로 회전한 2차배열을 다시 b[4][4]에 대입한다.

코드

void rotate(int b[4][4]){

int temp[4][4];

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

{

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

{

temp[i][j] = b[3 - j][i];

}

}

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

{

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

{

b[i][j] = temp[i][j];

}

}

//시계 방향으로 회전

}

2. Void draw\_board(int tot, int command)

이 함수에서는 score 즉 점수를 구하고, tot (total) 즉 보드에 나온 모든 수의 총합이 제대로 더해지는 확인한다.

Score을 구하는 것은 처음에 score를 0으로 초기화 해준 후 반복문을 통해 b[0][0] 부터 b[3][3] 까지의 값을 score 와 비교한다. 차례차례 비교하면서 그 값이 score보다 크면 score에 그 값을 대입한다.

함수를 호출하면 tot에 보드에 있는 모든 수의 합이 들어오는데 다시 보드의 있는 모든 수를 빼주면 tot는 0이 되어야 한다. 정말 0이되는지 확인하고 아니면 error를 출력한다.

코드

void draw\_board(int tot, int command){

int score=0;

/\* user code \*/

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

{

for (j = 0; j < 4; j++)

{

if (score < b[i][j])

score = b[i][j];

tot -= b[i][j];

}

}

if (tot != 0)

printf("error\n");

//판에서 가장 큰 수가 score , 여태까지 판에 나온 수를 전부 합한게 tot, tot가 제대로 되있는지 확인

/\* calculate score & check sum of all block equasls variable tot \*/

3. int is\_game\_over()

게임이 끝났을 경우는 모든 방향에 대해서 움직이지 않는 경우이다. 실제로 2차원 배열 b를 이동시키면 안되기 때문에 이동되는 것을 확인하기 위해 이차원 배열 b의 값을 저장해 둘 수 있는 배열 c를 만들고 c에 b를 복사한다. 그 후 모든 방향에 대해서 set\_board 함수를 호출하고 모든 방향에 대해서 움직이지 못하면 set\_board 함수는 계속 0을 반환할 것이다. 새로운 변수를 선언하여 4방향 모두 set\_board 함수를 실행하고 그 반환값의 합이 0이면 모든 방향에 대해서 움직이지 못한 것이므로 game over 이다. 그러므로 1을 반환해준다. 새로운 변수가 0이 아니면 game over가 아니므로 0을 반환해준다.

코드

int is\_game\_over(){

/\* user code \*/

int d=0;

int c[4][4];

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

{

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

{

c[i][j]=b[i][j];

}

}//b[4][4]를 c[4][4]에 복사

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

{

d=d+ set\_board(i,c);

}

if(d==0) return 1;

return 0;

//setboard를 불러와 모든 방향키를 다 대입해봐서 움직임이 없으면 game over gg bro

}

4. int make\_two\_or\_four()

이 함수에서 2또는 4가 나올 확률을 설정해준다. Srand(time(null)) 에 의하여 rand() 에 임의의 랜덤값이 들어가게 된다. Rand() 를 3으로 나눈 나머지는 0,1,2, 중 하나이고 0,1,2,가 나올 확률은 각각 3분의 1이다. 변수 input을 선언하여 rand%3이 0또는 1이면 input을 2라하고 rand%3이 2이면 input에 4를 대입한다. 이렇게 하면 2가 나올 확률은 3분의 2가 되고 4가 나올 확률은 3분의 1이 된다.

다음으로 2 또는 4가 출력될 장소를 설정해야 한다. 변수 num을 선언해서 배열 b에 있는 0의 개수를 세서 대입한다. 변수 count를 선언하고 rand()%num 한 값을 count에 대입한다. Count 에 어떠한 값이 들어갔는지는 모르지만 각 값이 나올 확률은 전부 같다. 따라서 이제 반복문을 이용해 배열 b에서 0을 찾은 다음 찾을때마다 count 값을 1씩 빼준다. Count 값이 0이 될 때 바로 그 0 자리에 input (2또는 4)를 대입하면 완벽하다.

코드

int make\_two\_or\_four(){

/\* user code \*/

int input,num=0,count;

input = rand() % 3;//input에 0,1,2 중 하나가 들어간다.

if (input == 1 || input == 0)

input = 2; //0,1,2 나올 확률이 각각 똑같으므로 input이 0또는 1이면 input=2이고 이는 3분의2확률이다.

else

input = 4; //0,1,2 나올 확률이 각각 똑같으므로 input이 2이면 input=4이고 이는 3분의 1 확률이다.

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

{

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

{

if (b[i][j] == 0)

num++;

}

}//판의 있는 0의 개수 = num

if (num == 0)

return 0;//0이 한개도 없으면 return 0;

count = rand() % num;

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

{

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

{

if (b[i][j] == 0)

{

if (count == 0)

{

b[i][j] = input;

return input;

}

count--;

}

}

}//판에 있는 임의의 0 자리에 2또는 4를 박는다.

/\* return 2 or 4 that makes in this times \*/

/\* if can not make two or four, then return 0 \*/

}

5. int set\_board(int dir, int b[4][4])

이 함수에서는 왼쪽으로 숫자를 이동시키고 더하는 것을 실행한다. 방향키가 왼쪽이 아니면 rotate 함수를 호출하여 회전시켜야 한다. 예를 들어 보자.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |

방향키 아래 키를 눌러서 1,2,3,4, 를 더한 후 맨 아래 쪽에 더한 값 10을 넣어야 한다.

그러려면 rotate 함수를 한번 호출해서

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 3 | 2 | 1 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

이렇게 만든다.

그 후 왼쪽으로 밀면서 합을 구하면

|  |  |  |  |
| --- | --- | --- | --- |
| 10 | 0 | 0 | 0 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

이렇게 된다. 회전을 한번 했으므로 다시 되돌리기 위해 시계방향으로 3번 더 회전하면

|  |  |  |  |
| --- | --- | --- | --- |
| 0 |  |  |  |
| 0 |  |  |  |
| 0 |  |  |  |
| 10 |  |  |  |

이렇게 되면서 아래방향으로 숫자들을 밀면서 합하게 되는 완벽한 상태가 된다.

이처럼 왼쪽 방향이 아닌 다른 방향키가 들어오면 적당히 회전한 해서 왼쪽으로 밀고 다시 원상복귀 하면 아름답게 밀리고 더해진다.

이번에도 c[4][4]를 만들어서 b[4][4] 를 똑같이 복사한 후에 대신 밀고 더할 것이다. 그리고 마지막에 c[4][4]와 b[4][4]를 비교해서 똑같으면 밀리고 더해지지 않은 것이요, 다르면 제대로 밀린 것이다. 제대로 밀렸으면 1을 return하고 밀리지 않았으면 0을 return 한다.

이제 밀고 더하는 것을 생각해보자.

우선 0 이 있으면 민다. 예를 들어 0 0 2 2 이면 2 2 0 0 이렇게 만든다. 0 2 2 2 이면 2 2 2 0 이렇게 만든다. 2 2 2 2 이면 밀지 않는다. 다음으로 더해 보도록 하자. 2 2 0 2 인 경우를 보도록 하자

2 2 0 2

왼쪽부터 읽어가면서 b[i][j]와 b[i][j+1]를 봐서 두개가 같으면 더해서 b[i][j]에 넣는다

4 2 0 2

그리고 b[i][j+2]를 b[i][j+1]로, b[i][j+3]을 b[j+2]로 옮겨 주고 b[i][j+3]에 0을 넣는다

4 0 2 0

더 민다

4 2 0 0

이런식으로 밀고 더한다.

코드

int set\_board(int dir, int b[4][4]) {

/\* user code init \*/

int i, j, k, sum = 0;

int num;

int c[4][4];

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

{

for (j = 0; j < 4; j++)

{

c[i][j] = b[i][j];

}

}//b[4][4]를 c[4][4]로 옮김

num=(dir+2)%4; //방향키에 따른 회전수

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

{

rotate(b);

} //방향키에 따른 회전

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

for (j = 0; j < 3; j++) {

if (b[i][j] == 0) {

for (k = j; k < 3 ; k++)

b[i][k] = b[i][k + 1];

b[i][3] = 0;

}

}

for (j = 0; j < 3; j++) {

if (b[i][j] == 0) {

for (k = j; k < 3 ; k++)

b[i][k] = b[i][k + 1];

b[i][3] = 0;

}

}

for (j = 0; j < 3; j++) {

if (b[i][j] == 0) {

for (k = j; k < 3 ; k++)

b[i][k] = b[i][k + 1];

b[i][3] = 0;

}

}

}//왼쪽으로 최대 3번 민다

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

for (j = 0; j < 3; j++) {

if ((b[i][j] == b[i][j + 1])) {

b[i][j] = b[i][j] + b[i][j + 1];

for (k = j+1; k < 3 ; k++) {

b[i][k] = b[i][k + 1];

}

b[i][3] = 0;

}

}

}//밀고 난 후 더한다.

for(i=4; i>num; i--)

{

rotate(b);

}//돌린거 다시 돌린다

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

{

for (j = 0; j < 4; j++)

{

if (c[i][j] != b[i][j])

return 1;

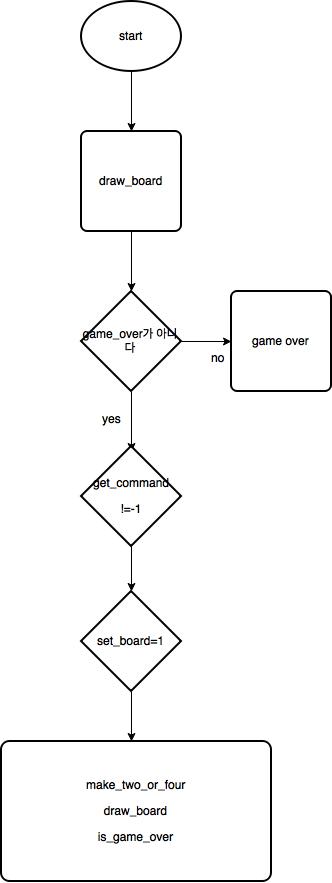
}

}//b와 c를 비교해서 밀고 합하는게 제대로 실행되었나 확인

return 0;

}

6. 실행 흐름도 int main(void)



대략적으로 flow chart를 그리면 이렇다. 구체적으로 메인함수를 설명하자. 처음에 tot를 0으로 초기화 시킨다. 게임은 2또는 4 2개로 시작하므로 make\_two\_or\_four 함수를 2번 호출하고 함수의 리턴값을 tot 더해 넣어준다. 다음으로 draw\_board 함수를 호출해준다

이제 게임오버가 될때까지 반복문을 반복한다. 방향키를 누르고 set\_board 함수를 호출하고 밀고 더할 수 있으면 다시 make\_two\_or\_four 함수를 호출해서 새로운 숫자 하나 써주고 draw\_board 실행하고 게임오버가 되는지 확인한다. 이것을 게임오버가 될 때 까지 반복한다.

코드

int main() {

int command = -1;

fp = fopen("output.txt", "w");

init();

srand(time(NULL));

int tot = 0;

/\* make init board \*/

/\* user code \*/

tot+=make\_two\_or\_four();

tot += make\_two\_or\_four();//tot에 판에 있는 총 합을 넣는다

draw\_board(tot, -1);

do {

command = GetCommand();

if (command != -1) {

if( set\_board(command, b) ){

/\* user code \*/

tot += make\_two\_or\_four();//tot에 판에 있는 총합을 넣는다

draw\_board(tot, command);

game\_over = is\_game\_over();//game over 를 확인

}

}

} while (!game\_over);

system("clear");

printf("game over!\n");

fclose( fp );

}