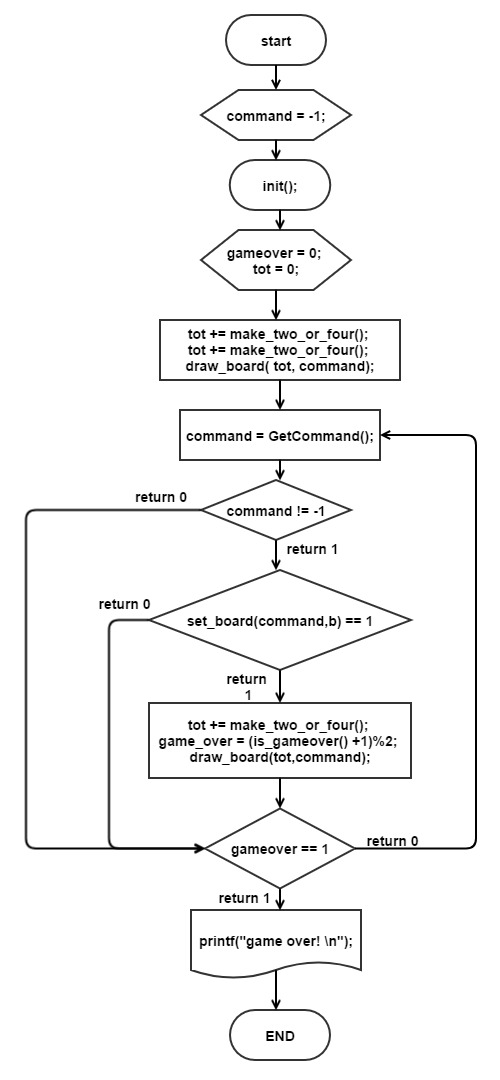
**2048 flow-chart 와 함수 설명 보고서**

**Main() 함수**

* 변수 초기 설정:

command = -1

“output.txt”에 write하기 위한 변수 fp를 초기화

tot = 0

* init()함수 호출
* srand(time(NULL)) 호출

rand()함수 호출할 때마다 다른 난수 생성하기 위한 호출.

main에서 호출했기 때문에 main에서 호출한 다른 함수 내에서도 rand()함수가 호출될 때마다 다른 난수가 생성된다.

* make\_two\_or\_four 함수 두 번 호출 후 draw\_board(tot,command)

make\_two\_or\_four 함수 호출로 전역변수 b[4][4] 내 임의의 두 칸에 2 또는 4를 채우고

칸을 채운 값이 반환되므로 그 반환값을 tot에 더한다.

그 후 draw\_board로 판을 그린다.

* GetCommand()함수의 반환값이 command

command는 유효한 키가 입력되었을 시에 0,1,2,3 중 하나의 값을 갖고

유효한 키가 입력되지 않았거나 오류가 있었을 시에 -1을 갖는다.

* command가 -1이면

game\_over가 1인지 아닌지 판단한다

game\_over가 1이면 “game\_over”가 프린트되고 아니면 다시 command를 GetCommand로 받는다.

* command가 -1이 아니면

set\_board(command,b)가 1인지 판단한다

* set\_board(command,b)가 0이면

game\_over가 1인지 아닌지 판단한다.

* set\_board(command, b)가 1이면

make\_two\_or\_four()호출로 임의의 빈 칸에 2 또는 4를 채우고 반환값(빈 칸을 채운 값)을 tot에 더한다

is\_game\_over로 게임이 끝났는 지 판단하고 그 결과에 따라 game\_over 값이 끝났으면 1, 안 끝났으면 0이 되도록 설정해준다

그리고 draw\_board(tot,command) 호출하여 판을 그린다

**void init() 함수**

* 전역변수 game\_over = 0 으로 초기화
* 전역변수 b[4][4]의 모든 칸을 0으로 초기화

**void rotate(int b[4][4])함수**

* 함수 내에서 다른 배열 temp[4][4]를 선언하여 b[4][4]를 돌리기 전 배열로 사용한다: temp[4][4]가 b[4][4]와 같게 만든다.
* 돌린 후의 b[i][j]의 값은 돌리기 전 b[3-j][i]의 값을 가짐을 이용해

b[i][j]=temp[3-j][i]로 b[4][4]를 시계방향으로 90도 돌린 배열을 b[4][4]에 넣는다.

**void set\_board(int dir, int b[4][4]) 함수**

* dir = (dir+3)%4
* 방향키가 위(w 또는 W)라고 정해놓고 칸을 옮기고 합치는 것으로 set\_board 함수를 짜놓았으므로

방향키에 따라 오른쪽으로 90도 돌리는 횟수를 계산한다.

* dir을 오른쪽으로 90도 돌리는 횟수로 변수의 의미를 바꾼다(= rotate 함수 호출 횟수)
* rotate함수에 b[4][4]배열을 parameter로 넣어 dir에 해당하는 값만큼 호출한다.
* 세로 한 줄씩 한 줄씩 수행하고 세로 한 줄마다 행 인덱스 1에서 시작해서 3으로의 순서로, 옮기고자 하는 칸을 변화시킨다.

-옮기고자 하는 칸( b[j][i] )이 어떻게 옮겨갈 지 수행하는 과정

cut이라는 변수 설정: b[j][i]가 옮겨가거나 합쳐질 수 없는 j인덱스를 가리킨다. 세로 한 줄 내에서 이미 합쳐진 칸이 없으면 cut은 -1로 초기화되어 있어야 하고 합쳐진 칸이 있으면 그 칸의 j인덱스가 cut의 값이 되어야한다.

count라는 변수 설정: set board 함수가 호출할 때마다 count값은 0으로 초기화되어야 되고 옮김이 있을 경우를 센다.

1. b[j][i]가 0인 경우: 옮길 필요가 없으니 다음 칸 b[j+1][i]에 대해 조건 만족하는 지 알아본다.
2. b[j-1][i] (b[j][i]의 앞칸)가 cut인 경우(j-1 == cut): 옮길 수 없으므로 다음 칸에 대해 조건 만족하는 지 알아본다.
3. b[j-1][i] (b[j][i]의 앞칸)이 0인 경우:

옮길 수 있으므로 아래 어떤 수행을 하건 count+=1을 시행한다.

b[j-k][i] 를 k=2 부터 k를 1씩 키우면서 다음 조건 만족하는 지 알아보기. 단 그러다가 다음 조건을 만족하는 경우가 있으면, 해당 작업 후 다음 칸 b[j+1][i]에 대해 처음에 했던 조건 만족하는 지 알아본다.

1. b[j-k][i]가 cut인 경우: b[j-k+1][i] = b[j][i] 하고 b[j][i] =0
2. b[j-k][i]가 cut은 아니면서 b[j][i]와 같은 값인 경우: b[j-k][i]를 2\*b[j][i] 값으로 바꾸고 b[j][i]를 0으로 만든 다음 cut을 j-k로 설정한다.
3. b[j-k][i]가 앞의 경우가 아니면서 0이 아닌 값을 갖는 경우: b[j-k+1][i]를 b[j][i]값으로 바꾸고 b[j][i]로 바꾼다.
4. b[j-1][i] (b[j][i]의 앞칸)가 cut은 아니면서 b[j][i]와 같은 경우: b[j-1][i]를 b[j][i] \* 2의 값으로 바꾸고 b[j][i]는 0으로 바꾼다. 그리고 cut=j-1로 설정한다. 옮길 수 있으므로 count+=1 시행한다. 다음 칸에 대해 조건 만족하는 지 알아본다.
5. (else에 해당되는 경우) b[j-1][i] (b[j][i]의 앞칸)가 cut도 아니고 b[j][i]와 다르고 0이 아닌 값을 갖는 경우: 수행없이(옮김 없이) 다음 칸에 대해 조건 만족하는 지 알아본다.

* 옮기고 합치는 작업으로 b[4][4]를 변화시킨 후 rotate함수에 b[4][4]를 parameter로 넣고 4-dir만큼 호출시킨다.(= 오른쪽으로 90도 돌리는 작업을 4-dir만큼 수행한다)
* count = 0이면 옮김이 없었다는 의미로 0을 반환시킨다.
* count != 0이면 옮김이 있었다는 의미로 1을 반환시킨다.

**int is\_game\_over()함수**

* 함수 내에서 set\_board를 호출하기 전마다 함수 내에서 선언한 temp[4][4] 배열을 parameter로 들어온 b[4][4]와 같게 만든 후에, set\_board함수의 parameter로 temp[4][4]를 넣고, parameter의 command 자리에는 0,1,2,3 순서로 넣는다. 그런데 시행 중간에 set\_board의 반환값으로 1이 나오면 옮길 수 있다는 말로 game over가 아니라는 반환값 1을 내보낸다. 시행하는 내내 set\_board의 반환값이 0이면 어떤 방향으로 움직이건 옮김이 없었다는 뜻으로 game over가 되었다는 반환값 0을 내보낸다.

**void draw\_board(int tot, int command)함수**

* score을 0으로 시작해서 b[4][4]의 모든 칸의 값과 score 값을 비교하는 데 score 값보다 큰 값을 비교하게 되면 score값을 그 값으로 바꾼다. 그래서 score값은 b[4][4]의 모든 값 중 최댓값을 갖게 된다. 그리고 parameter로 들어온 tot의 값에서 b[4][4]의 모든 값을 빼서 0이 되지 않으면 오류를 출력한다.
* 화면에 score 값을 표시하고 output.txt에 score과 command와 b[4][4]의 모든 값을 써놓는다. 그리고 숫자에 색깔을 입혀서 b[4][4]의 모든 값을 화면에 출력한다.

**int make\_two\_or\_four() 함수**

* rand()함수는 모든 수를 같은 확률로 생성해내므로 rand()%3의 결과값이 될 수 있는 0,1,2는 모두 생성될 확률이 같다. rand()%3이 0 또는 2일 때 (rand()%3)%2의 결과값이 0이 되고, rand()%3이 1일 때 (rand()%3)%2의 결과값이 1이 된다. 따라서 (rand()%3)%2의 결과값으로 0이 나올 확률은 2/3이고 결과값으로 1이 나올 확률은 1/3이다. 이를 이용해서 randomnum = (((rand()%3)%2)+1)\*2하면 randomnum이 2가 될 확률은 2/3이고 randomnum이 4가 될 확률은 1/3이다.
* i=0부터 3까지 변화시키는 데 i에 대해서 j=0부터 3까지 변화시키면서 b[i][j]가 0인 칸이 있으면 exist=1이 되어 빈공간이 존재함을 알아낸다. 그러나 b[i][j]가 0인 칸이 없으면 exist=0으로 고정되면서 빈공간이 없음의 반환값 0을 반환한다.
* rand()%4의 결과값 0,1,2,3이 나올 확률은 모두 같다.
* exist가 1이면 행 인덱스 i를 rand()%4로 0,1,2,3을 랜덤하게 지정하고 j를 rand()%4로 0,1,2,3을 랜덤하게 지정하는 시행을 b[i][j]가 0일 때까지 시행한다. (빈공간이 있을 경우 시행한 것으로 무한히 돌아가지는 않는다.) 그 b[i][j]에 randomnum 값을 넣는다. 반환값으로 randomnum을 반환한다.

**int getch(void)함수**

* 사용자로부터 입력받은 키를 int형으로 반환해준다.

**int GetCommand() 함수**

* getch()로부터 입력받은 키가 무엇인지 받아 ch에 넣는다.
* ch가 ‘d’ 또는 ‘D’이면 0을 반환한다.

ch가 ‘w’ 또는 ‘W’이면 1을 반환한다.

ch가 ‘a’ 또는 ‘A’이면 2를 반환한다.

ch가 ‘s’ 또는 ‘S’이면 3을 반환한다.

* switch문을 들어가지 않는 오류가 발생했거나 ch가 ‘d’, ’D’, ’w’, ’W’, ’a’, ’A’, ’s’, ’S’ 중에 하나가 아니면 -1을 반환한다.