#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <termios.h>//read from the console

#define SIZE 16

#define exit0 0x1B

#define UP 0x57

#define UP\_B 0x77

#define DOWN 0x53

#define DOWN\_B 0x73

#define RIGHT 0x64

#define RIGHT\_B 0x27

#define LEFT 0x41

#define LEFT\_B 0x61//define directions

int arr[SIZE] = {0};

int is\_move = 0; //define the move yes or no

int is\_merge = 0; //define the merge yes or no

int max; //the max number

int total; //the score

static struct termios oldt;

void restore\_terminal\_settings(void)

{

tcsetattr(0, TCSANOW, &oldt);

}

//set return value

void disable\_terminal\_return(void)

{

struct termios newt;

tcgetattr(0, &oldt);

newt = oldt;

newt.c\_lflag &= ~(ICANON | ECHO);

tcsetattr(0, TCSANOW, &newt);

atexit(restore\_terminal\_settings);

}

void init(void)

{

int i, random1, random2;

random1 = rand() % SIZE;

for(i = 0; i < 1; i++)

{

random2 = rand() % SIZE;

if(random1 == random2)

{

i--;

}

}

arr[random1] = 2;

arr[random2] = 2;

}

//initialize data

int is\_dead()

{

int i, j;

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

{

if(!arr[i])

{

return 0;

}

}

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

{

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

{

if(arr[j] == arr[j + 1])

{

return 0;

}

}

}

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

{

for(j = i; j < i + 12; j += 4)

{

if(arr[j] == arr[j + 4])

{

return 0;

}

}

}

return 1;

}

//define the game can continue to play or not

//if all of 16 are numbers and cant merge ,you loss

int max\_num(void)

{

int i;

static int count = 0;

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

{

if(arr[i] > max)

{

max = arr[i];

}

}

if(!count && 2048 == max)

{

count++;

printf("congratulation!you win!");

getchar();

}

return max;

}

//if 2048 is existed ,game over ,you win

void print\_game(void)

{

system("clear");

int i;

printf("退出游戏:esc\n操作:wasd\n最大的数是：%d\n总分是：%d\n", max\_num(), total);

printf("\n");

printf("+------------- 2048 --------------+\n");

printf("+-----------------------------------+\n");

printf("| | | | |\n");

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

{

if(i == 0 || i == 4 || i == 8 || i == 12)

{

printf("|");

}

printf(0 == arr[i] ? "%9c|" : "%8d|", arr[i]);//if 0 print blank

if(i == 3 || i == 7 || i == 11)

{

printf("\n");

printf("|--------+--------+--------+--------|\n");

printf("| | | | |\n");

}

}

printf("\n");

printf("+-----------------------------------+\n");

printf("\n\n\n");

if(is\_dead())//if is\_dead ,you loss and initialize data again

{

int i;

printf("game over\n");

printf("请按任意键重新开始\n");

getchar();

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

{

arr[i] = 0;

}

init();

print\_game();

}

}

void rand\_num(void)

{

while(is\_move || is\_merge)

{

int random = rand() % SIZE;

if(!arr[random])

{

arr[random] = 2;

break;

}

}

is\_move = 0;

is\_merge = 0;

}

//after moving ,you will get new 2 nunber at blank

void move\_go(int loop\_count, int current\_i, int direction)

{

int i;

for(i = 0; i < loop\_count; i++)

{

if(arr[current\_i] && !arr[current\_i + direction])

{

arr[current\_i + direction] = arr[current\_i];

arr[current\_i] = 0;

current\_i += direction;

is\_move = 1;

}

}

}

//three parameter:1.the cycle-index 2.the moving number 3.the direction

//moving function

void move\_up(void)

{

int i, loop\_count, direction;

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

{

if(arr[i])

{

loop\_count = i / 4;

direction = -4;

move\_go(loop\_count, i, direction);

}

}

}

void move\_down(void)

{

int i, loop\_count, direction;

for(i = SIZE - 1; i >= 0; i--)

{

if(arr[i])

{

loop\_count = (4 - 1) - i / 4;

direction = 4;

move\_go(loop\_count, i, direction);

}

}

}

void move\_right(void)

{

int i, loop\_count, direction;

for(i = SIZE - 1; i >= 0; i--)

{

if(arr[i])

{

loop\_count = (4 - 1) - (i + 4) % 4;

direction = 1;

move\_go(loop\_count, i, direction);

}

}

}

void move\_left(void)

{

int i, loop\_count, direction;

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

{

if(arr[i])

{

loop\_count = (i + 4) % 4;

direction = -1;

move\_go(loop\_count, i, direction);

}

}

}

//four moving function of four direction ,they will use the move\_go direction to

//make their move

void merge(int current\_i, int direction)

{

if(arr[current\_i] && arr[current\_i + direction] && arr[current\_i] == arr[current\_i + direction])

{

arr[current\_i] = arr[current\_i + direction] \* 2;

total += arr[current\_i];

arr[current\_i + direction] = 0;

//current\_i += direction;

is\_merge = 1;

}

}

//the function of merge ,finish one merge

void move\_up\_pre(void)

{

move\_up();

int i, j, direction = 4;

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

{

for(j = i; j < i + 12; j += 4)

{

merge(j, direction);

}

}

move\_up();

}

void move\_down\_pre(void)

{

move\_down();

int i, j, direction = -4;

for(i = 4 - 1; i >= 0; i--)

{

for(j = i + 12; j >= 4; j -= 4)

{

merge(j, direction);

}

}

move\_down();

}

void move\_right\_pre(void)

{

move\_right();

int i, j, direction = -1;

for(i = 4 - 1; i >= 0; i--)

{

for(j = 4 \* i + 3; j > 4 \* i; j--)

{

merge(j, direction);

}

}

move\_right();

}

void move\_left\_pre(void)

{

move\_left();

int i, j, direction = 1;

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

{

for(j = 4 \* i; j < 4 \* i + 3; j++)

{

merge(j, direction);

}

}

move\_left();

}

//four functions of four directions ,the moving and delete the blank

int main(void)

{

srand(time(NULL));

init();

print\_game();

disable\_terminal\_return();

while(1)

{

switch(getchar())

{

case exit0:

exit (0);

case UP:

case UP\_B:

move\_up\_pre();

rand\_num();

print\_game();

break;

case DOWN:

case DOWN\_B:

move\_down\_pre();

rand\_num();

print\_game();

break;

case RIGHT:

case RIGHT\_B:

move\_right\_pre();

rand\_num();

print\_game();

break;

case LEFT:

case LEFT\_B:

move\_left\_pre();

rand\_num();

print\_game();

break;

default:

break;

}

}

return 0;

}

//the main