\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Report: HW7

Author: F74024729 林冠圻 <lltzpp@gmail.com>

Class: 資訊系112級 乙班

Description:

安排地雷方法：random 行與列，假設該位置已被設置炸彈則重新 random

dfs Function: 開啟空白格，對該點做深度優先搜尋法，不斷往八個方向的點擴展，直到該點並非空白格

calc function: 計算每個格子的數字應該為多少

input function: 讀取兩個數字

output function: 輸出結果

check function: 判斷這張地圖是否完成，如果所有非地雷的格子都是已開啟，那麼就是已完成

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Code:

#include<stdio.h>

#include<time.h>

#include<stdlib.h>

#define Nmax 30

int map[Nmax + 5][Nmax + 5], used[Nmax + 5][Nmax + 5], X, Y, i, j, res;

void dfs ( int x, int y ){

int i, j;

if ( \*( \*( used + x ) + y ) || \*( \*( map + x ) + y ) == -2 ) return;

if ( \*( \*( map + x ) + y ) >= 0 ) \*( \*( used + x ) + y ) = 1;

if ( \*( \*( map + x ) + y ) ) return;

for ( i = -1 ; i < 2 ; i++ ) for ( j = -1 ; j < 2 ; j++ ) if ( i || j ) dfs ( x + i, y + j );

}

int check ( int n ){

for ( i = 1 ; i < n ; i++ ) for ( j = 1 ; j < n ; j++ ) if ( !\*( \*( used + i ) + j ) && 0 <= \*( \*( map + i ) + j ) ) return 1;

return 0;

}

int calc ( int x, int y ){

for ( i = -1, res = 0 ; i < 2 ; i++ ) for ( j = -1 ; j < 2 ; j++ ) !~\*( \*( map + x + i ) + y + j ) && ( i || j ) ? res++ : res;

return res;

}

void input ( void ){scanf ( "%d %d", &X, &Y );}

void output ( int n ){for ( i = 1 ; i < n ; i++, putchar ( '\n' ) ) for ( j = 1 ; j < n ; j++ ) printf ( "%c ", ( \*( \*( used + i ) + j ) ) ? ( !\*( \*( map + i ) + j ) ) ? '\_' : '0' + \*( \*( map + i ) + j ) : '?' );}

int main ( int argc, char \*argv[] ){

srand ( clock() + time ( NULL ) );

int n = atoi ( argv[1] ), m = atoi ( argv[2] ), i, j, N = n++, x = rand() % N + 1, y = rand() % N + 1;

for ( i = 0 ; i <= n ; i++ ) for ( j = 0 ; j <= n ; j++ ) \*( \*( map + i ) + j ) = -2, \*( \*( used + i ) + j ) = 1;

for ( i = 1 ; i < n ; i++ ) for ( j = 1 ; j < n ; j++ ) \*( \*( map + i ) + j ) = \*( \*( used + i ) + j ) = 0;

while ( m-- && ( \*( \*( map + x ) + y ) = -1 ) ) while ( \*( \*( map + x ) + y ) == -1 ) x = rand() % N + 1, y = rand() % N + 1;

for ( i = 1 ; i < n ; i++ ) for ( j = 1 ; j < n ; j++ ) \*( \*( map + i ) + j ) = ( \*( \*( map + i ) + j ) < 0 ? -1 : calc ( i, j ) );

while ( check ( n ) ){

system ( "clear" ), output ( n ), input();

if ( !~\*( \*( map + X + 1 ) + Y + 1 ) ) return puts ( "you are dead" ), 0;

if ( !\*( \*( map + X + 1 ) + Y + 1 ) ) dfs ( X + 1, Y + 1 );

else if ( \*( \*( map + X + 1 ) + Y + 1 ) > 0 ) \*( \*( used + X + 1 ) + Y + 1 ) = 1;

}

system ( "clear" ), output ( n ), puts ( "you win" );

}

Compilation:

gcc -o mine hw6.c -lm

Execution:

./mine 4 4

Output:

? ? ? ?

? ? ? ?

? ? ? ?

? ? ? ?

0 0

\_ \_ 1 ?

\_ \_ 1 ?

\_ \_ 1 1

\_ \_ \_ \_

0 3

\_ \_ 1 1

\_ \_ 1 ?

\_ \_ 1 1

\_ \_ \_ \_

you win