**Lab4**

1. **恶意输入防范。**

主要通过cin.clear();cin.ignore(1024,'\n');来清楚错误状态和缓冲区输入字符，防止程序崩溃。当用户输入不合法时，提示用户的输入不满足要求，并要求用户重新输入。



1. 右键标记功能。同时在每次用户输入前，实时打印用户已标记的地雷的数量和剩余数量，给用户做出提示。

**void** map::mark(**int** i, **int** j){//标记地雷

**if**(hmp[i][j]=='#'){

hmp[i][j]='@';

mnumber++;

rnumber--;

**return**;

}

**if**(hmp[i][j]=='@'){

hmp[i][j]='#';

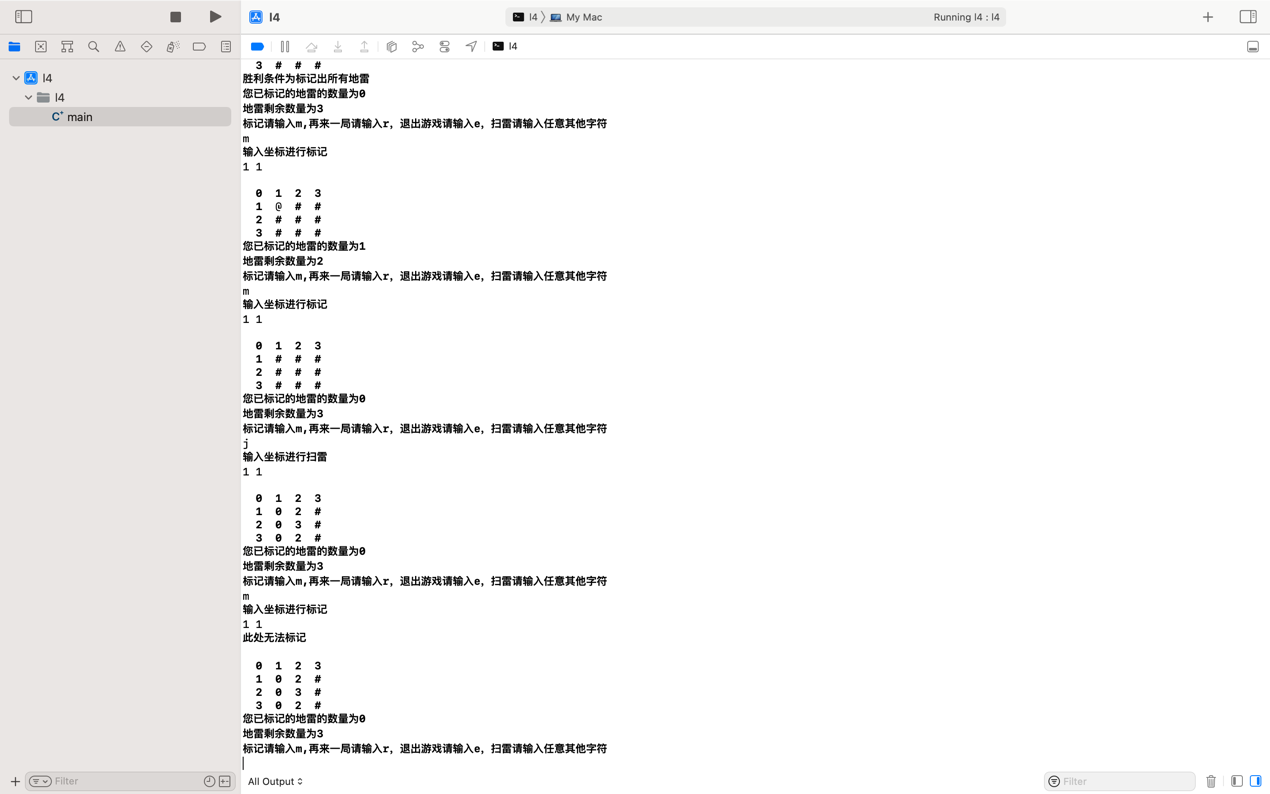
mnumber--;

rnumber++;

**return**;

}

cout<<"此处无法标记"<<endl;

}

1. 保证用户首次选择的位置不会是地雷。通过设置一个变量f=1，在用户输入第一个坐标后再生成地雷图（除去此位置），然后将f设为0，避免一次游戏中反复更新地雷图。

4.当点击的位置周边地雷数量为零时，可以自动扩大显示的范围，将周边不是地雷的点全部展示出来。若揭开一处位置为0，则使用两个函数相互调用的方式扩大揭开其周围八个位置，同时，一旦揭开后有数字（大于等于1），则此处不再扩大范围。

**void** map::search(**int** i, **int** j){//自动扩大显示的范围

hmp[i][j]=qmp[i][j];

**if**(qmp[i][j]=='0'){

**for**(**int** s=i-1; s<i+2; s++){

**for**(**int** t=j-1; t<j+2; t++){

**if**(hmp[s][t]=='#')

hmp[s][t]=snumber(s,t);

}

}

}

}

**char** map::snumber(**int** i, **int** j){//自动扩大显示的范围

**if**(qmp[i][j]=='0')

search(i,j);

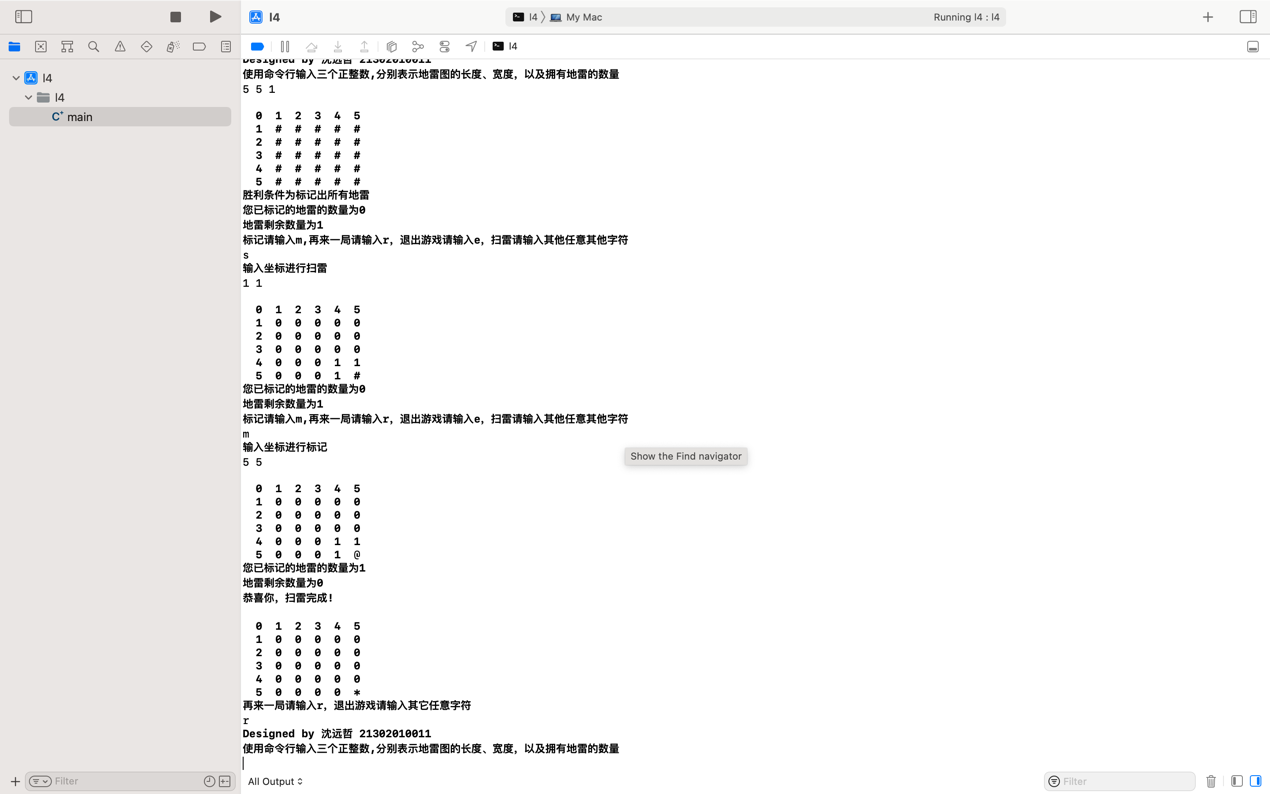
**return** qmp[i][j];

}



5.在第二条中已一并论述。

6. 当用户在游戏中或本轮游戏结束后，利用return main（）在不退出程序的情况下，重新开始一轮新的游戏。





7.若扫雷地图较大，每次行和列数都要数一次会比较麻烦，所以在打印地雷图时将行和列数打出，方便输入准确位置。

**void** map::print(**char** (&pmp)[102][102] ){//打印地雷图方阵

cout<<endl;

**for**(**int** i=0; i<col+1; i++)

cout<<setw(3)<<i;

cout<<endl;

**for**(**int** i=1; i<row+1; i++){

cout<<setw(3)<<i;

**for**(**int** j=1; j<col+1; j++){

cout<<setw(3)<<pmp[i][j];

}

cout<<endl;

}

}

总结：这四次实验锻炼了我的逻辑思维能力与资料收集能力（善用搜索）。在lab4中，我主要遇到了两个难点。其一是在防范恶意输入时，通过了解缓冲区的原理和部分针对缓冲区操作的函数解决了部分输入会导致程序陷入死循环的难题。其二是设计自动扩大显示的范围时碰到的算法问题，如使用单一函数递归导致揭开的数量及位置不正确，两个函数相互调用出现死循环等问题。后来通过对地雷图位置是否已经揭开进行判断解决了死循环的问题。几次的代码修改与重构让我明白了程序不仅仅需要能运行，更需要有扩展性，需要注重细节排除bug来使程序稳定运行。