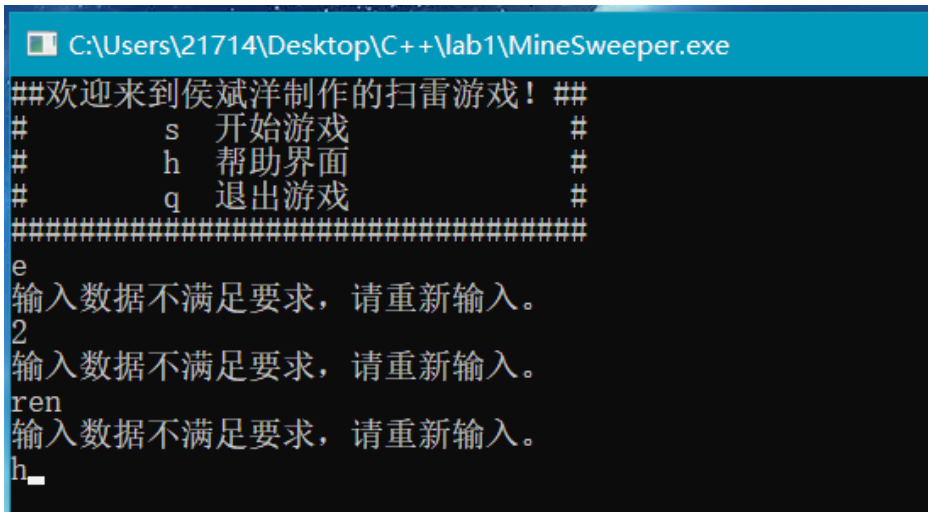
Lab3

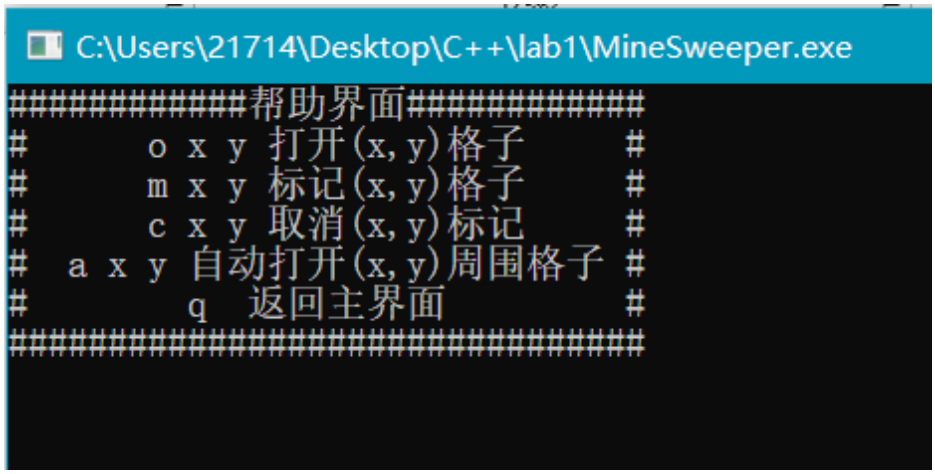
1：运行结果

（1）进入程序出现主界面

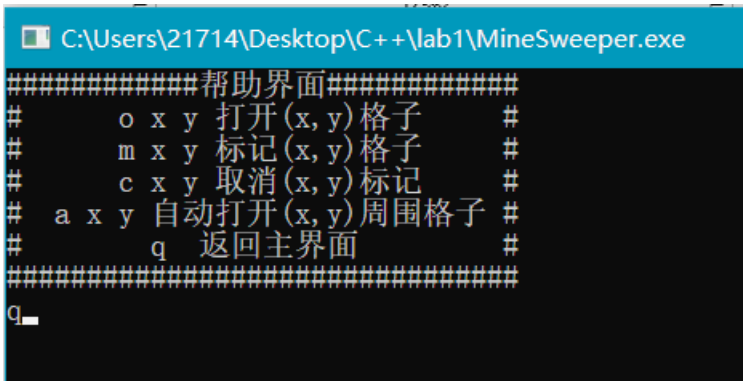
（2）在主界面做了恶意输入防范

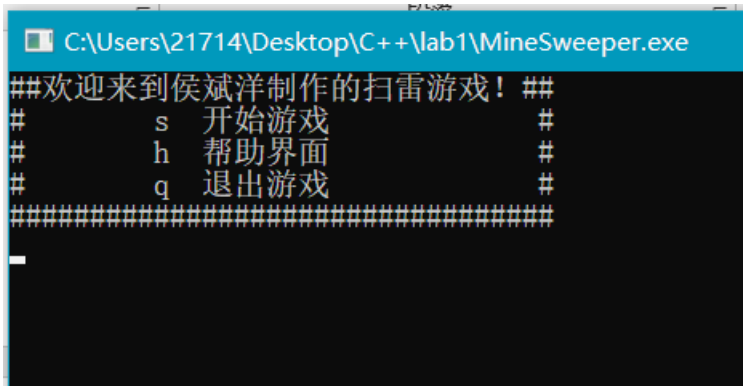


（3）输入 h 进入帮助界面

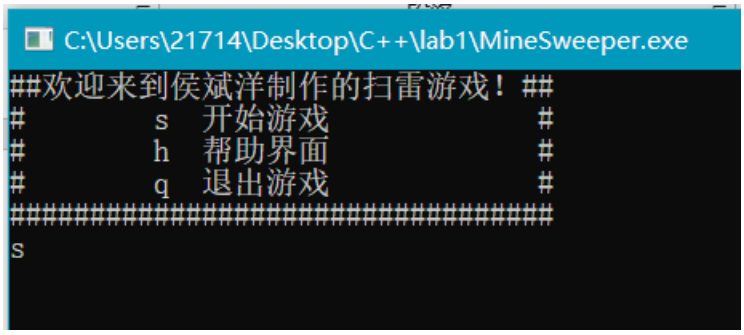


（4）之后按 q 返回主界面

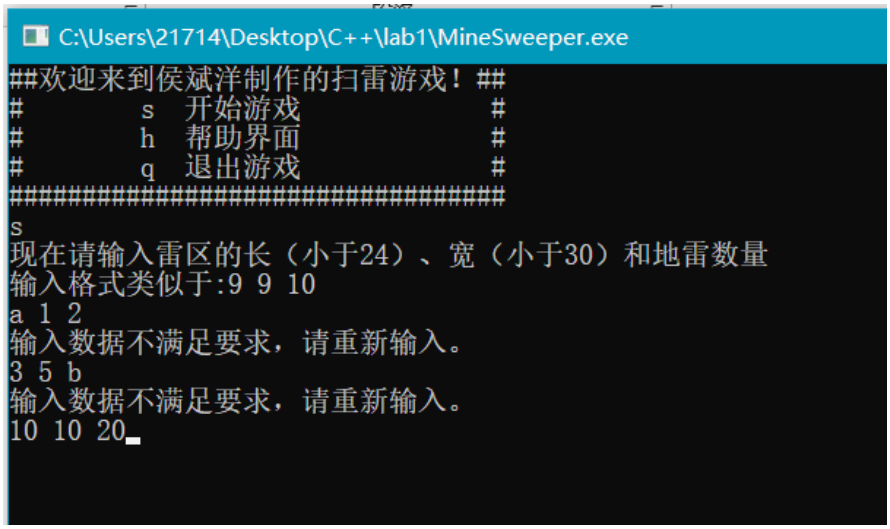




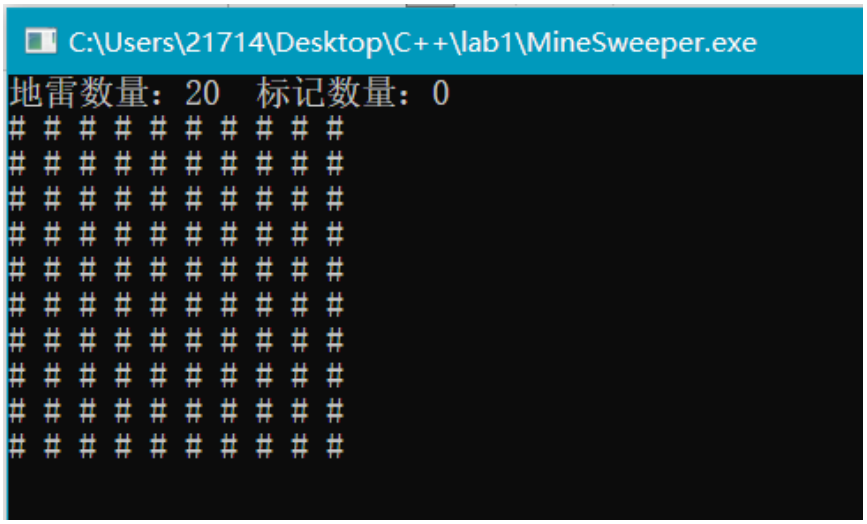
（5）然后按 s 开始游戏



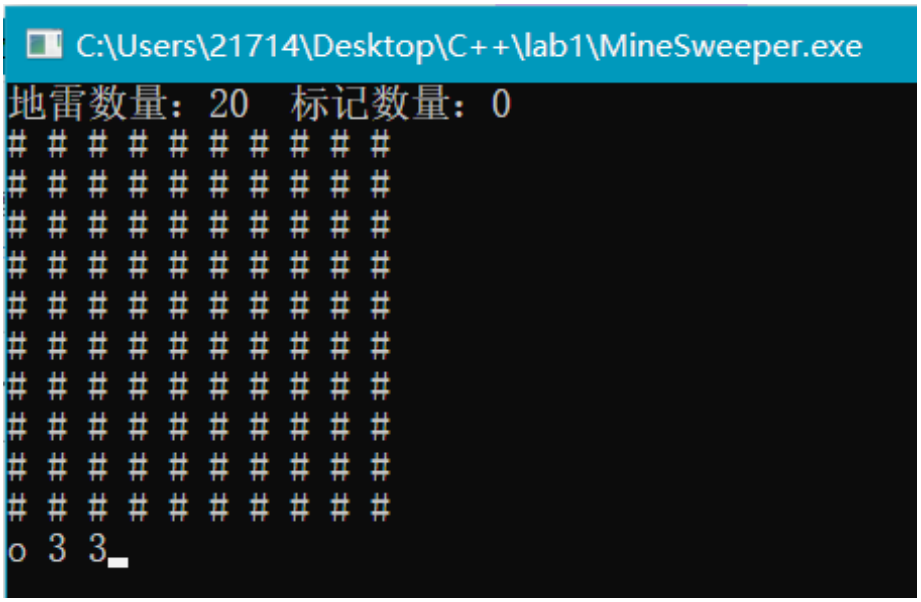
（6）在输入地雷参数时也做了恶意输入防范

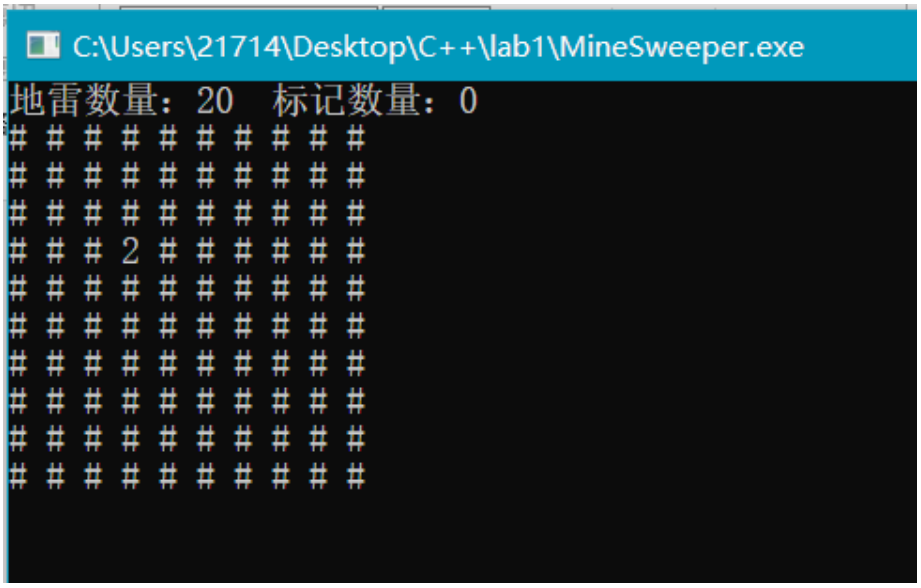


（7）输入正确数据后打印地雷图



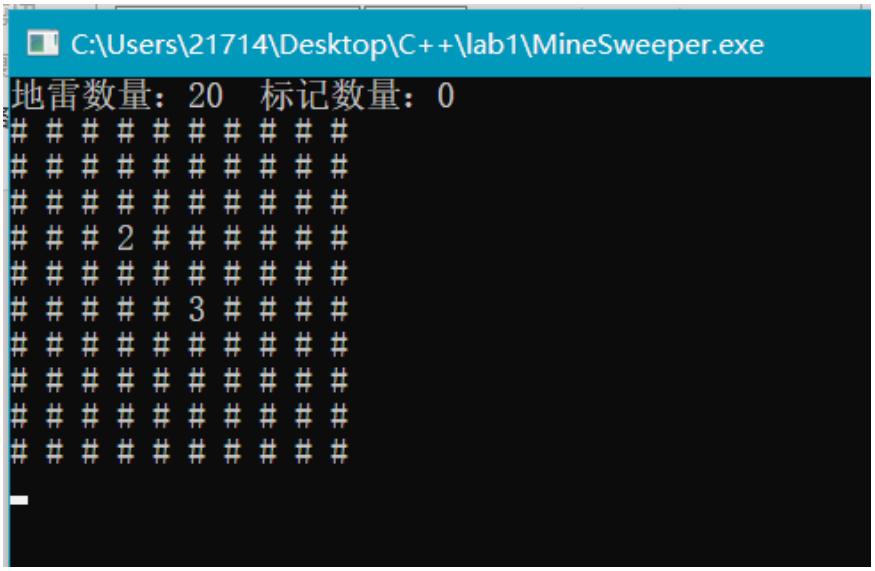
（8）输入 o 3 3 指令打开坐标为（3,3）的格子。（注：此处保证第一次打开必不为地雷。）



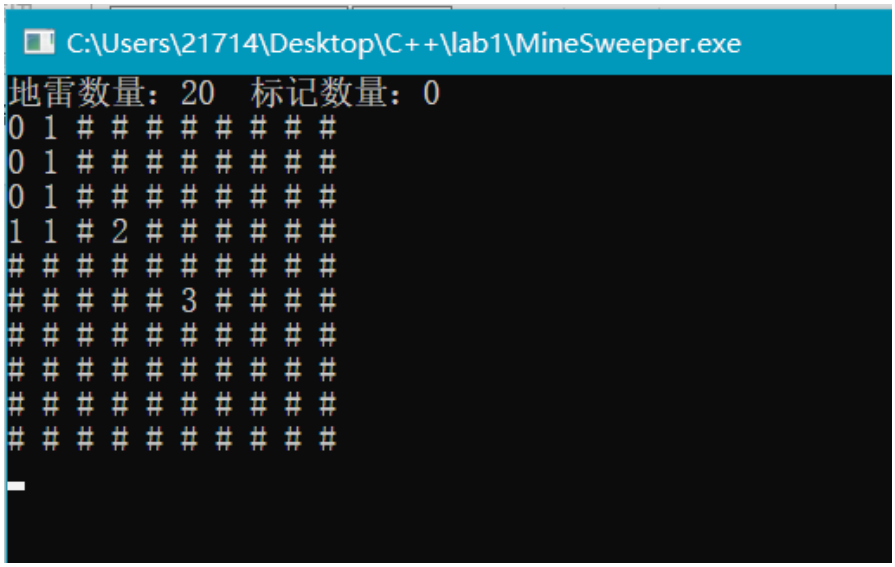


（9）然后输入 a 5 5 指令自动打开（5,5）及其周边无地雷的格子

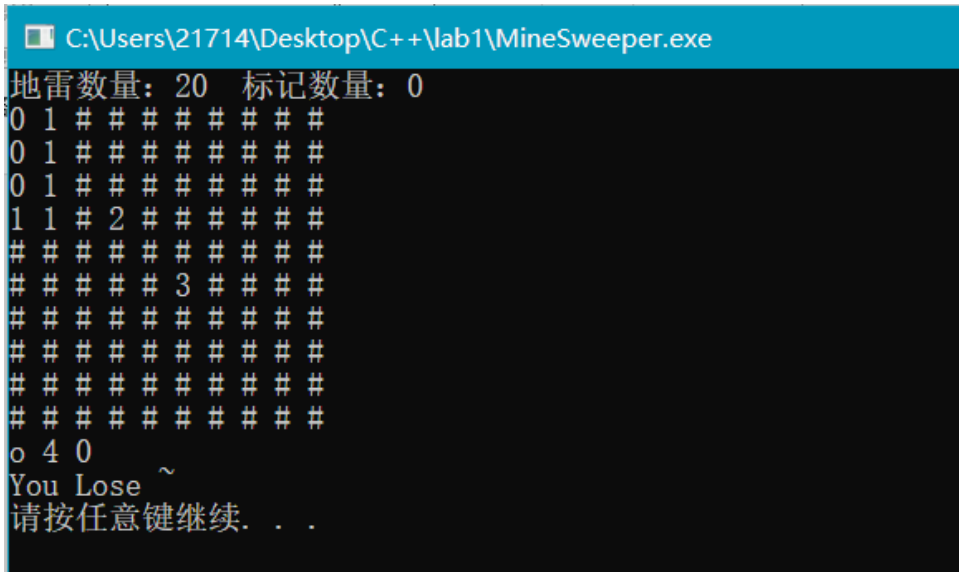
（10）这里因为（5,5）周围有地雷，故只打开了一个格子。



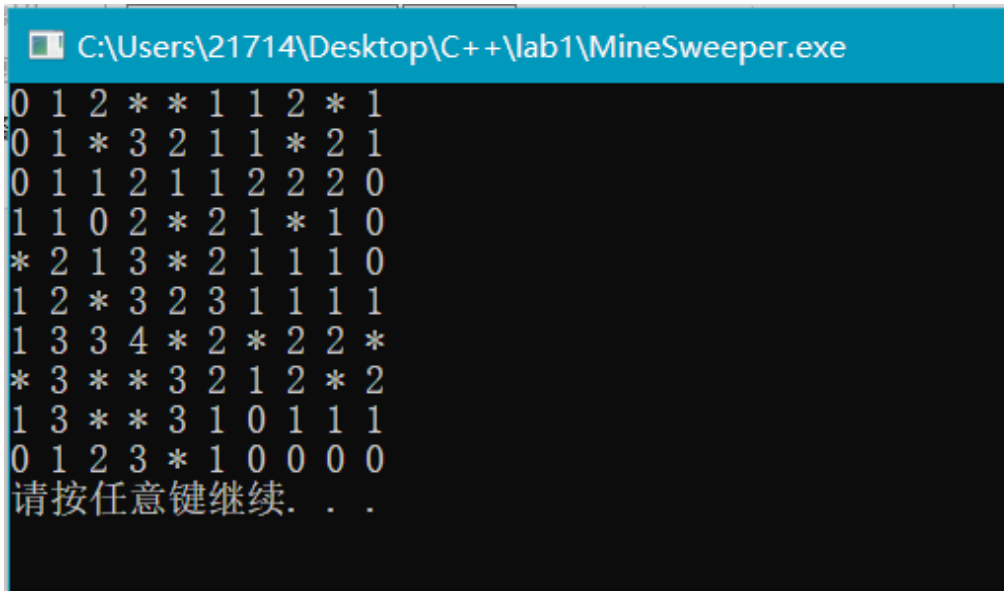
（11）然后输入指令 a 0 0 则出现以下结果



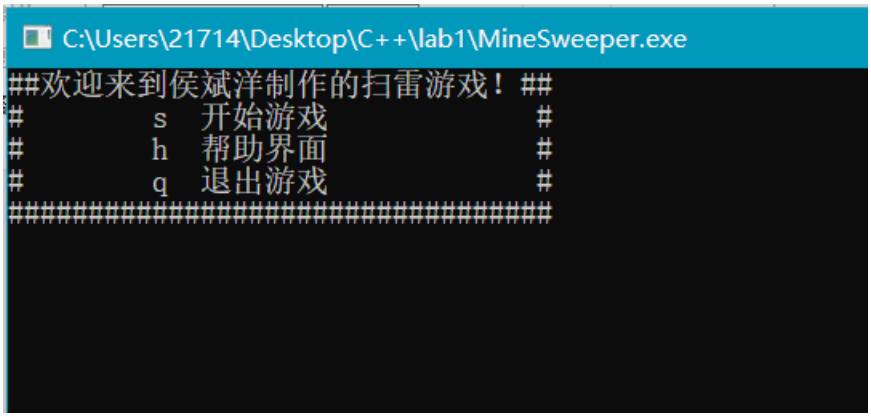
（12）接下来输入 o 4 0（验证失败案例），(4,0）处恰为地雷，故判断为失败。



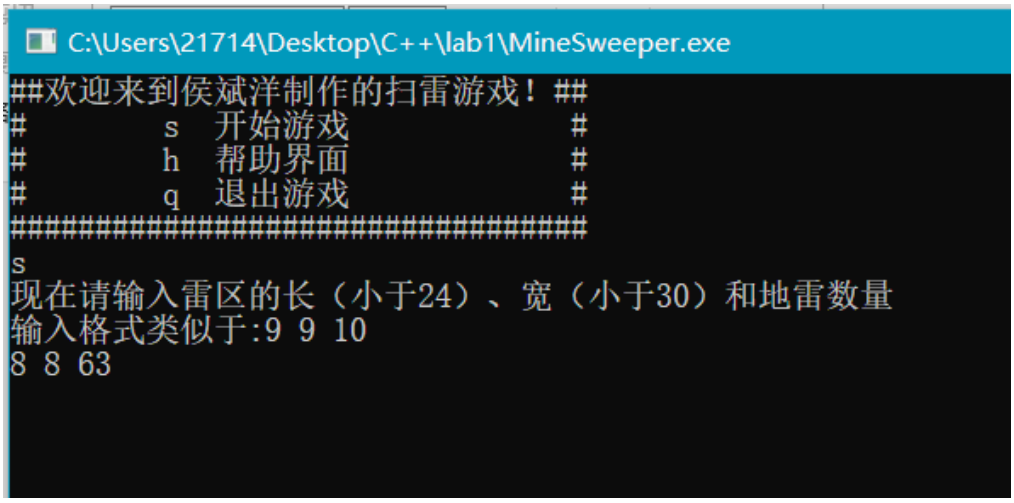
（13）按任意键后打印完整地雷图，从下图也可以看出（4，0）处确实为地雷。



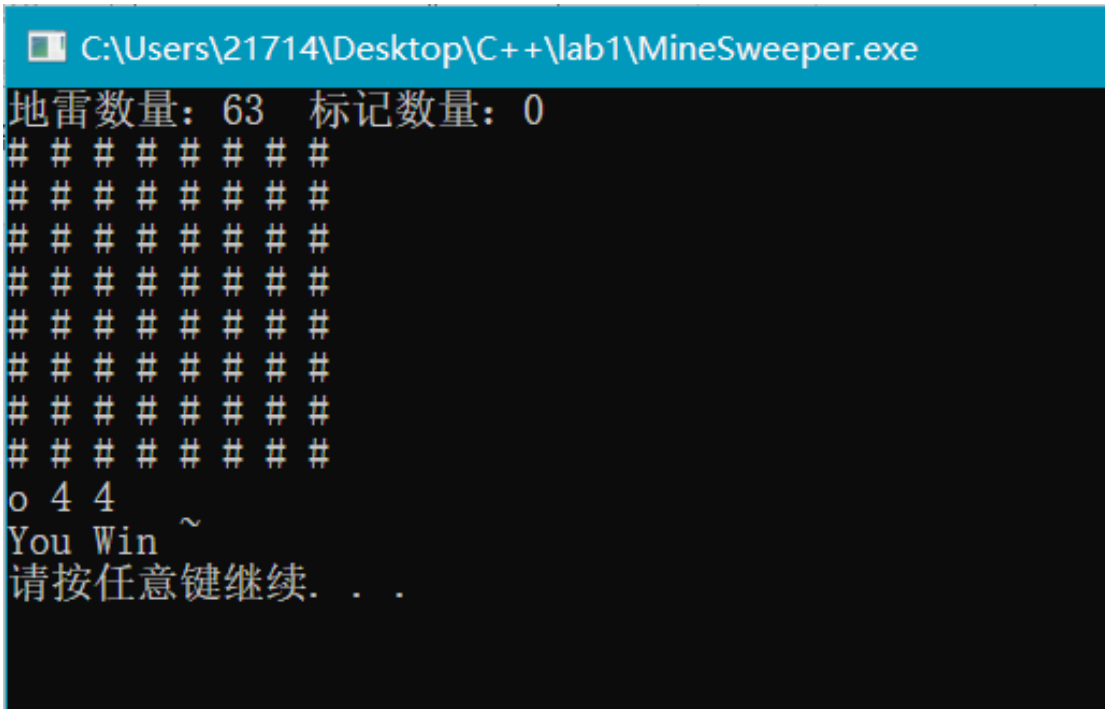
（14）再按任意键返回主界面。（注：游戏中直接按 q 也可以返回主界面）

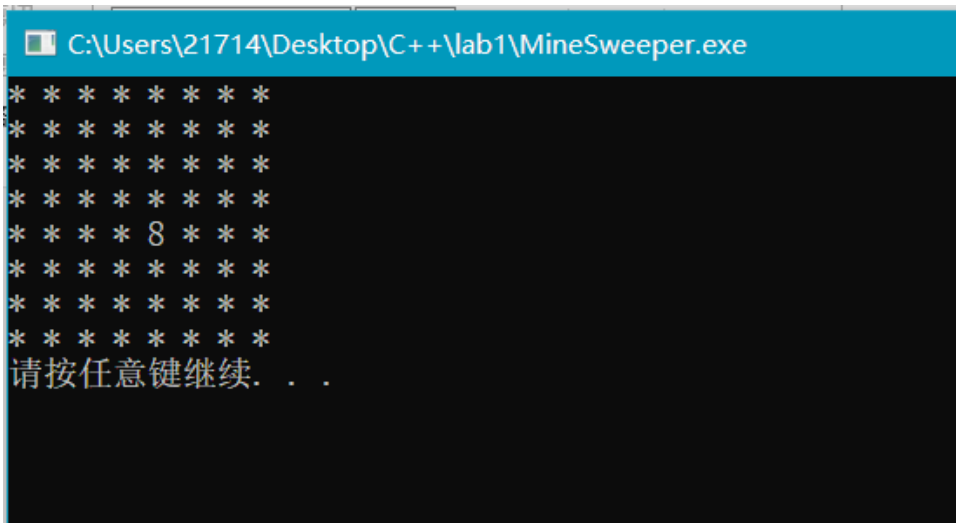


（15）继续进行测试（验证一个特殊的成功案例，同时验证第一个格子必不为雷）

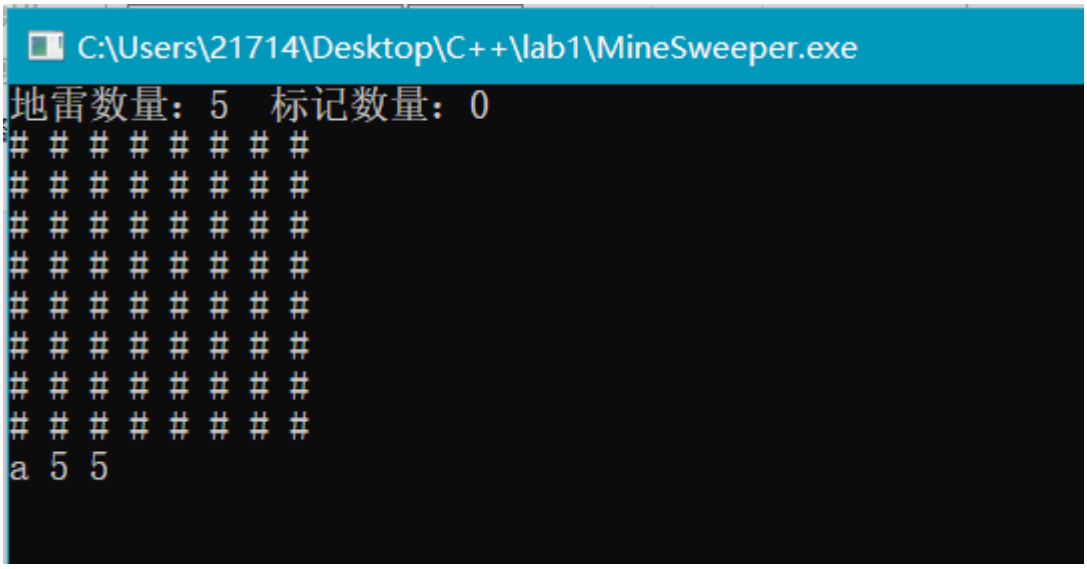


（16）由于只有一个雷且第一次必不为雷，故直接胜利

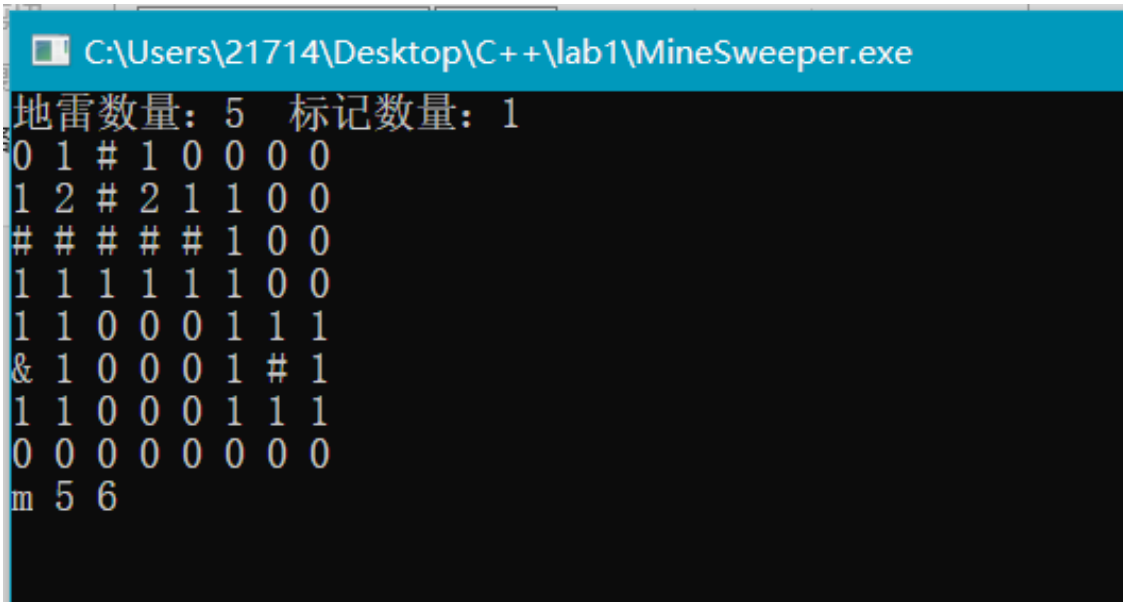




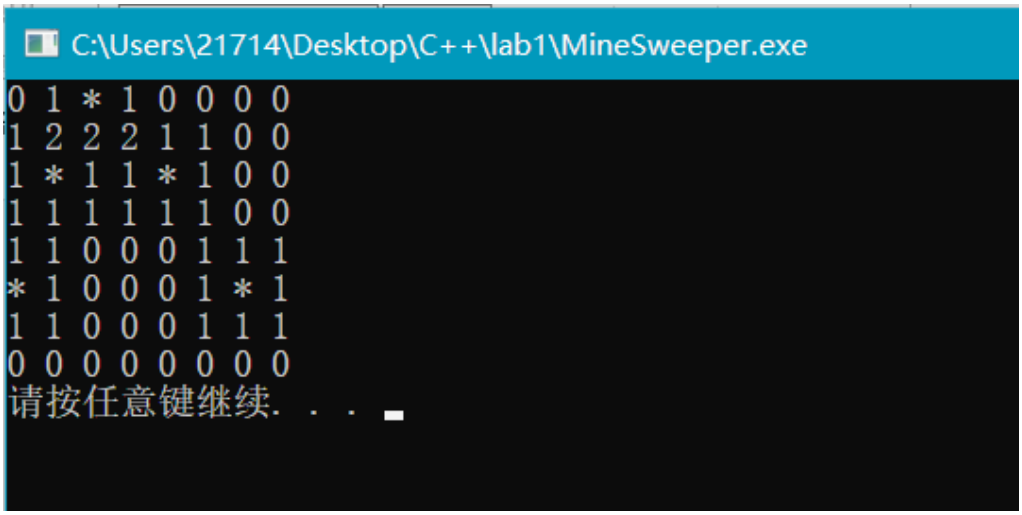
（17）然后验证普遍情况下的成功案例



（18）下图展示了标记的操作：输入 m 5 6 标记（5,6）位置，并在右上角更新标记数量。

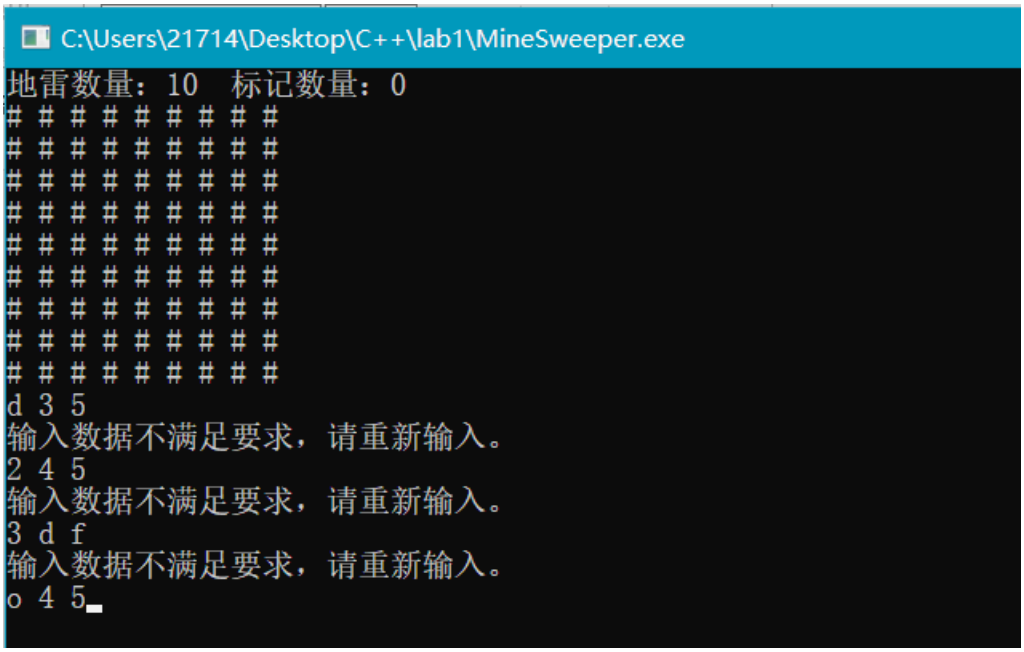


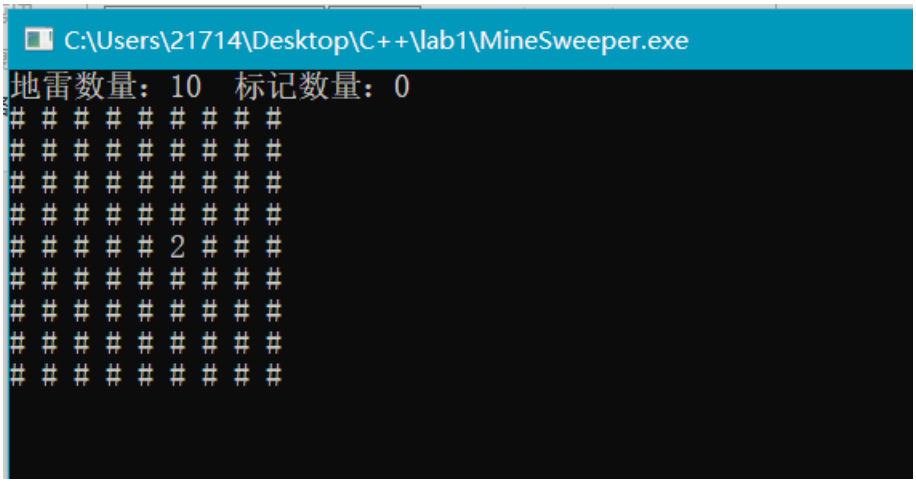
（19）下图是一般情况下的成功案例



（20）此外，在游戏过程中也做了恶意输入防范

（注：在游戏的全部输入过程中都做了恶意输入防范，提示进行正确的输入）





2：代码原理

（1）首先定义一个类名为 MineMap（地雷图），其中 private 包含了地雷图的长，宽，地雷数量，以及地雷图二维数组的指针，辅助标记图二维数组的指针；public 包含了有参构造函数，无参构造函数，析构函数，复制构造函数，重载赋值运算符，以及游戏过程中需要对地雷图执行的各种操作函数。同时注意要用#ifndef 语句防止重复定义（在 MineMap.h 文件中）如下：

```
#ifndef MINEMAP_H_
#define MINEMAP_H_

class MineMap
{
private:
    int m_length;
    int m_width;
    int m_MineNum;

    //对于地雷图，-1表示有雷，其余数字表示周围雷的数量
    int **m_map;

    //对于标记图，1表示该位置已被打开，10表示被标记，0表示未被打开也未被标记
    int **m_mark;
public:
    MineMap();
    MineMap(int length, int width, int MineNum);
    ~MineMap();
    MineMap(const MineMap &copy);
    MineMap &operator=(const MineMap &copy);
    void Open(int x, int y);
    void AutoOpen(int x, int y);
    void Mark(int x, int y);
    void CancelMark(int x, int y);
    void ShowMap();
    void Endshow();
    int JudgeWin();
    int JudgeLose();
};

#endif
```

（2）然后就是编写这些函数的代码。其中无参构造函数默认地雷图长、宽和地雷数量为9,9,10。有参构造函数需要输入数据。在构造函数中包含了生成随机地雷图的算法，以保证初始化的每个对象都是随机的。由于 private 中存在二维数组，故也定义了复制构造函数，重载了赋值运算符以备用，同时注意在析构函数中用双层 for 循环来 delete 二维数组。然后其他成员函数在注释中都包含了其对应的功能，具体实现细节也较为简单。不过 AutoOpen 函数的实现还是有一定难度的，我这里用了深度优先的算法，但用广度优先的话也是可以的，不过深度优先代码简洁一些。其实这里如果用哨兵算法的话可以省去很多判断操作，代码会更加简洁，但如果用哨兵算法的话我这里就需要改很多东西。（在 MineMap.cpp 文件中）。如下：

```
/*****************************MineMap成员函数*********************************/
//无参构造函数
MineMap::MineMap()…

//有参构造函数
MineMap::MineMap(int length, int width, int MineNum)…

//析构函数
MineMap::~MineMap()…

//复制构造函数
MineMap::MineMap(const MineMap &copy)…

//重载赋值运算符
MineMap &MineMap::operator=(const MineMap &copy)…

//打开单个格子
void MineMap::Open(int x, int y)…

//自动打开周围格子
void MineMap::AutoOpen(int x, int y)…

//标记地雷
void MineMap::Mark(int x, int y)…

//取消标记
void MineMap::CancelMark(int x, int y)…

//展示表层地雷图
void MineMap::ShowMap()…

//展示底层地雷图
void MineMap::Endshow()…

//判断是否成功，成功返回1，继续则返回0.
int MineMap::JudgeWin()…

//判断是否失败，失败返回-1，继续则返回0.
int MineMap::JudgeLose()…
/*****************************************************************************/
```

```
/*********************************其他函数************************************/
using namespace std;

//打印主界面
void PrintMain()…

//打印帮助界面
void PrintHelp()…

/*****************************************************************************/
```

（3）最后就是组织主程序了，组织主程序的主要思路为：①while 循环控制程序的进行和界面之间的跳转（界面跳转可能会用到一些标签变量作为 while 循环判断条件）；②由于程序在同一阶段下可能会有不同的输入，因而在主程序中用 switch 语句来对输入进行选择并根据输入执行不同的操作；③在不同的功能区合理安排函数的位置以实现内部数据的处理，同时建立一个较为整洁的用户界面（可以用 system 函数清屏等操作整理界面）；④根据程序需要对用户的输入进行引导，同时防范恶意输入；⑤对于每个功能区，存在一些只用函数难以解决的问题，解决这些问题并对细节进行优化，同时做好注释。（在 MineSweeper.cpp 文件中）

以下是一些具体的思路:
一、游戏总体思路：二维数组 m_map 存放地雷图，若某处为地雷则设为-1，若不为地雷则用其他数字表示周围的地雷数量。二维数组 m_mark 存放游戏过程中的操作，初始全为 0，若打开了某个格子则设为 1；标记某个格子则设为 10；取消标记则设为 0（因为对已经打开的格子进行标记是没有意义的）。根据 m_map 和 m_mark 对应位置的关系就可以打印游戏过程中的地雷图，并判断游戏的成功失败情况。

二、以下是一个恶意输入防范的例子，以此例子来说明恶意输入防范的思路

```
//在主界面中读取命令
cin >> Order;

/*****************输入检验******************/
while (cin.fail() || (Order != 's' && Order != 'h' && Order != 'q'))
{
    cout << "输入数据不满足要求，请重新输入。" << endl;
    cin.clear();
    while (cin.get() != '\n')
    {
        continue;
    }
    cin >> Order;
}
/*****************************************/
```

cin 在读取输入后，若输入类型不匹配，则 cin.fail()的值就设为 1，这样就防止了不同类型的恶意输入。然后就只需考虑逻辑上的恶意输入，这里利用判断条件使得程序只接受给出的三个值中的一个，这样也解决了逻辑上的恶意输入，保证了输入的正确性。同时要注意 cin 缓冲区的问题，在下一次输入之前要清除错误状态（cin.clear()）并清空缓冲区（while 循环）。总而言之：①防范类型输入错误；②防范逻辑输入错误；③清空错误状态并重新输入。

三、在本次 lab 中我区分了主界面和游戏界面，并且还做了帮助界面（见上述运行结果），这样在进行一局游戏后可以直接回到主界面进行选择，并且可以查看帮助界面以了解我所定义的游戏操作，另外在对地雷图参数的输入中也对用户进行了引导，这样可以使用户快速上手，不会出现一头雾水的情况。

四、以下代码保证了第一次打开格子不为地雷

```
//使第一次打开的格子不为地雷
while (lose == -1 && first == 0)
{
    map = MineMap(length, width, MineNum);
    map.Open(x, y);
    lose = map.JudgeLose();
    win = map.JudgeWin();
    if (lose == 0)
    {
        first = 1;
    }
}
first = 1;
```

其中 first 是在外部定义的一个 int 型变量，初始值为 0；当外部循环执行第一次时，first 的值为 0，因此可能会进入这个循环，此时进行讨论：①第一次打开的格子为雷，则 lose==-1，进入循环，此时重新创造一个对象并将其赋给 map（这里就用到了上面定义过的复制构造函数和重载的赋值运算符），循环一直进行直到新的地图在这个格子上没有地雷，此时将 first 设为 1 结束循环，之后 first 无论经过多少次循环都为 1；因此不会再进入这个循环，防止该循环对正常游戏进行干扰；②第一次打开格子不为雷，则本次不会进入循环，而之后 first 的值又立马被设为 1；故之后的过程中也不会再进入这个循环。

五、以下代码实现自动打开周围格子

```
//自动打开周围格子
void MineMap::AutoOpen(int x, int y)
{
    m_mark[x][y] = 1;
    if (m_map[x][y] == 0)
    {
        static int a[3] = {-1, 0, 1};
        static int b[3] = {-1, 0, 1};
        for (int m = 0; m < 3; m++)
        {
            for (int n = 0; n < 3; n++)
            {
                if (x + a[m] >= 0 && x + a[m] < m_length && y + b[n] >= 0 && y + b[n] < m_width
                && m_mark[x + a[m]][y + b[n]] != 10 && m_mark[x + a[m]][y + b[n]] != 1)
                {
                    m_mark[x + a[m]][y + b[n]] = 1;
                    MineMap::AutoOpen(x + a[m], y + b[n]);
                }
            }
        }
    }
}
```

用深度优先搜索找出 m_map 中为 0 且 m_mark 中不为 1（这步判断十分重要，可以防止函数一直递归下去）和 10 的格子并打开其周围的格子完成自动打开格子操作。

3：代码附录

（1）无参构造函数和有参构造函数中含有随机数算法，在 lab1 中已列出，因篇幅较长，本次不列出，在 lab4 中也可以直接看代码。而复制构造函数、析构函数和重载赋值运算符都是一些套路代码。

```
//析构函数
MineMap::~MineMap()
{
    for (int i = 0; i < m_length; i++)
    {
        delete[] m_map[i];
        delete[] m_mark[i];
    }
    delete[] m_map;
    delete[] m_mark;
}
```

```
//复制构造函数
MineMap::MineMap(const MineMap &copy)
{
    m_length = copy.m_length;
    m_width = copy.m_width;
    m_MineNum = copy.m_MineNum;
    m_map = new int *[m_length];
    for (int i = 0; i < m_length; i++)
    {
        m_map[i] = new int[m_width];
    }
    for (int i = 0; i < m_length; i++)
    {
        for (int j = 0; j < m_width; j++)
        {
            m_map[i][j] = copy.m_map[i][j];
        }
    }

    m_mark = new int *[m_length];
    for (int i = 0; i < m_length; i++)
    {
        m_mark[i] = new int[m_width];
    }
    for (int i = 0; i < m_length; i++)
    {
        for (int j = 0; j < m_width; j++)
        {
            m_mark[i][j] = copy.m_map[i][j];
        }
    }
}
```

```cpp
//重载赋值运算符
MineMap &MineMap::operator=(const MineMap &copy)
{
    if (this == &copy)
    {
        return *this;
    }
    m_length = copy.m_length;
    m_width = copy.m_width;
    m_MineNum = copy.m_MineNum;

    for (int i = 0; i < m_length; i++)
    {
        delete[] m_map[i];
        delete[] m_mark[i];
    }
    delete[] m_map;
    delete[] m_mark;

    m_map = new int *[m_length];
    for (int i = 0; i < m_length; i++)
    {
        m_map[i] = new int[m_width];
    }
    for (int i = 0; i < m_length; i++)
    {
        for (int j = 0; j < m_width; j++)
        {
            m_map[i][j] = copy.m_map[i][j];
        }
    }

    m_mark = new int *[m_length];
    for (int i = 0; i < m_length; i++)
    {
        m_mark[i] = new int[m_width];
    }
    for (int i = 0; i < m_length; i++)
    {
        for (int j = 0; j < m_width; j++)
        {
            m_mark[i][j] = copy.m_mark[i][j];
        }
    }
    return *this;
}
```

（2）其他成员函数（AutoOpen 函数见上述）

```cpp
//打开单个格子
void MineMap::Open(int x, int y)
{
    if (m_mark[x][y] != 10)
        m_mark[x][y] = 1;
}
```

```cpp
//标记地雷
void MineMap::Mark(int x, int y)
{
    m_mark[x][y] = 10;
}
```

```cpp
//取消标记
void MineMap::CancelMark(int x, int y)
{
    m_mark[x][y] = 0;
}
```

```cpp
//展示表层地雷图
void MineMap::ShowMap()
{
    int count = 0;
    for (int i = 0; i < m_length; i++)
    {
        for (int j = 0; j < m_width; j++)
        {
            if (m_mark[i][j] == 10)
                count++;
        }
    }
    std::cout << "地雷数量：" << m_MineNum << "  标记数量：" << count << std::endl;
    for (int i = 0; i < m_length; i++)
    {
        for (int j = 0; j < m_width; j++)
        {
            if (m_mark[i][j] == 0)
            {
                std::cout << '#' << ' ';
            }
            else if (m_mark[i][j] == 10)
            {
                std::cout << '&' << ' ';
            }
            else if (m_mark[i][j] == 1)
            {
                std::cout << m_map[i][j] << ' ';
            }
        }
        std::cout << std::endl;
    }
}
```

```cpp
//展示底层地雷图
void MineMap::Endshow()
{
    for (int i = 0; i < m_length; i++)
    {
        for (int j = 0; j < m_width; j++)
        {
            if (m_map[i][j] == -1)
            {
                std::cout << '*' << ' ';
            }
            else
            {
                std::cout << m_map[i][j] << ' ';
            }
        }
        std::cout << std::endl;
    }
}
```

```cpp
//判断是否成功，成功返回1，继续则返回0.
int MineMap::JudgeWin()
{
    int count = 0;
    for (int i = 0; i < m_length; i++)
    {
        for (int j = 0; j < m_width; j++)
        {
            if (m_map[i][j] != -1 && m_mark[i][j] == 1)
                count++;
        }
    }
    if (count == m_length * m_width - m_MineNum)
        return 1;
    else
        return 0;
}
```

```cpp
//判断是否失败，失败返回-1，继续则返回0.
int MineMap::JudgeLose()
{
    for (int i = 0; i < m_length; i++)
    {
        for (int j = 0; j < m_width; j++)
        {
            if (m_map[i][j] == -1 && m_mark[i][j] == 1)
                return -1;
        }
    }
    return 0;
}
```

注：这里代码虽不整齐，但在 cmd 上的输出是整齐的。

```cpp
//打印主界面
void PrintMain()
{
    cout << "##欢迎来到侯斌洋制作的扫雷游戏！##" << endl;
    cout << "#         s   开始游戏              #" << endl;
    cout << "#         h   帮助界面              #" << endl;
    cout << "#         q   退出游戏              #" << endl;
    cout << "####################################" << endl;
}
```

```
//打印帮助界面
void PrintHelp()
{
    cout << "#############帮助界面#############" << endl;
    cout << "#       o x y 打开(x,y)格子      #" << endl;
    cout << "#       m x y 标记(x,y)格子      #" << endl;
    cout << "#       c x y 取消(x,y)标记      #" << endl;
    cout << "#  a x y 自动打开(x,y)周围格子 #" << endl;
    cout << "#         q  返回主界面          #" << endl;
    cout << "###################################" << endl;
}
```

（3）主程序中的一些细节（考虑到用图片看那么长的程序不太方便）。以下主要是游戏过程的一些代码并且以下几张图片是上下相连的

```
//创建一个随机地雷图对象map
MineMap map = MineMap(length, width, MineNum);
map.ShowMap();

char order;              //存放游戏界面的命令
int x, y;                //存放进行操作的坐标
int first = 0;           //标记第一次打开某个格子的操作，使第一次打开的格子不为地雷
int win = 0, lose = 0;   //标记游戏的成功与失败

//读取游戏界面的命令
cin >> order;

/*****************输入检验******************/
while (cin.fail() || (order != 'q' && order != 'o' && order != 'm' && order != 'c' && order != 'a'&&
order!='x'))
{
    cout << "输入数据不满足要求，请重新输入。" << endl;
    cin.clear();
    while (cin.get() != '\n')
    {
        continue;
    }
    cin >> order;
}
/*****************************************/
```

```
/*****************输入检验******************/
while (cin.fail() || (order != 'q' && order != 'o' && order != 'm' && order != 'c' && order != 'a'&&order!='x'))
{
    cout << "输入数据不满足要求，请重新输入。" << endl;
    cin.clear();
    while (cin.get() != '\n')
    {
        continue;
    }
    cin >> order;
}
/*****************************************/
```

```
//游戏界面内的循环，控制本次游戏的进行
while (order != 'q') // quit
{
    //读取进行操作的坐标
    cin >> x >> y;

    /*****************输入检验*****************/
    while (x >= length || y >= width || cin.fail() || x < 0 || y < 0)
    {
        cout << "输入坐标不满足要求,请重新输入x和y。" << endl;
        cin.clear();
        while (cin.get() != '\n')
        {
            continue;
        }
        cin >> x >> y;
    }
    /*****************************************/

    //对游戏界面的命令进行选择
    switch (order)
    {

    //打开单个格子
    case 'o': // open
    {
        map.Open(x, y);

        //判断游戏成功失败情况
        // lose==-1失败，win==1成功，lose==win==0继续
        lose = map.JudgeLose();
        win = map.JudgeWin();
```

```
//使第一次打开的格子不为地雷
while (lose == -1 && first == 0)
{
    map = MineMap(length, width, MineNum);
    map.Open(x, y);
    lose = map.JudgeLose();
    win = map.JudgeWin();
    if (lose == 0)
    {
        first = 1;
    }
}
first = 1;

//游戏继续进行
if (lose == 0 && win == 0)
{
    system("CLS");
    map.ShowMap();
    cin >> order;

    /*****************输入检验*****************/
    while (cin.fail() || (order != 'q' && order != 'o' && order != 'm' && order != 'c' && order != 'a'))
    {
        cout << "输入数据不满足要求，请重新输入。" << endl;
        cin.clear();
        while (cin.get() != '\n')
        {
            continue;
        }
        cin >> order;
    }
    /*****************************************/
}
//游戏失败
```

```
//游戏失败
else if (lose == -1)
{
    cout << "You Lose ~\n";
    system("Pause");

    //给玩家展示完整地雷图
    system("CLS");
    map.Endshow();
    system("Pause");

    //自动返回主界面
    order = 'q';
}
//游戏成功
else if (win == 1)
{
    cout << "You Win ~\n";
    system("Pause");

    //给玩家展示完整地雷图
    system("CLS");
    map.Endshow();
    system("Pause");

    //自动返回主界面
    order = 'q';
}
break;
```

```
//标记地雷
case 'm': // mark
{
    map.Mark(x, y);
    system("CLS");
    map.ShowMap();
    cin >> order;

    /****************输入检验*****************/
    while (cin.fail() || (order != 'q' && order != 'o' && order != 'm' && order != 'c' && order != 'a'))
    {
        cout << "输入数据不满足要求，请重新输入。" << endl;
        cin.clear();
        while (cin.get() != '\n')
        {
            continue;
        }
        cin >> order;
    }
    /*****************************************/

    break;
}
```

```
//标记地雷
case 'm': // mark
{
    map.Mark(x, y);
    system("CLS");
    map.ShowMap();
    cin >> order;

    /****************输入检验*****************/
    while (cin.fail() || (order != 'q' && order != 'o' && order != 'm' && order != 'c' && order != 'a'))
    {
        cout << "输入数据不满足要求，请重新输入。" << endl;
        cin.clear();
        while (cin.get() != '\n')
        {
            continue;
        }
        cin >> order;
    }
    /***************************************/

    break;
}
```

```
//取消标记
case 'c': // cancel
{
    map.CancelMark(x, y);
    system("CLS");
    map.ShowMap();
    cin >> order;

    /****************输入检验*****************/
    while (cin.fail() || (order != 'q' && order != 'o' && order != 'm' && order != 'c' && order != 'a'))
    {
        cout << "输入数据不满足要求，请重新输入。" << endl;
        cin.clear();
        while (cin.get() != '\n')
        {
            continue;
        }
        cin >> order;
    }
    /***************************************/

    break;
}
```