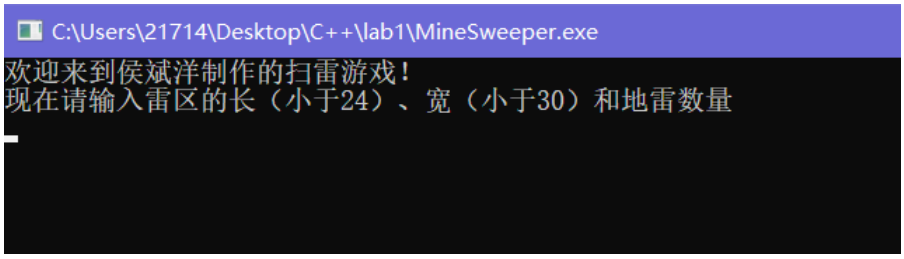
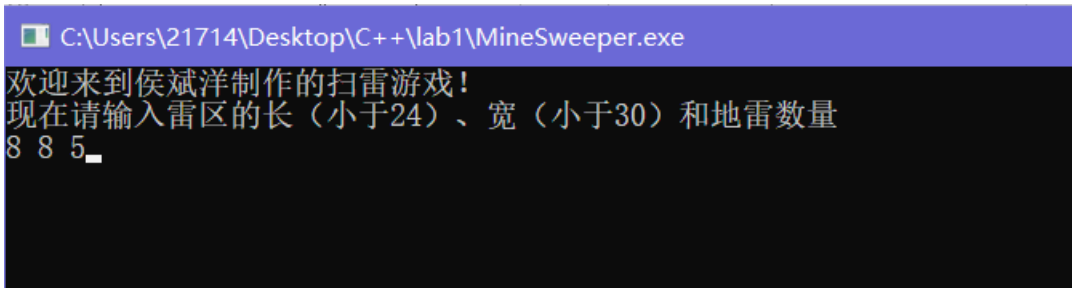
Lab1

1:运行结果截图

① 进入程序

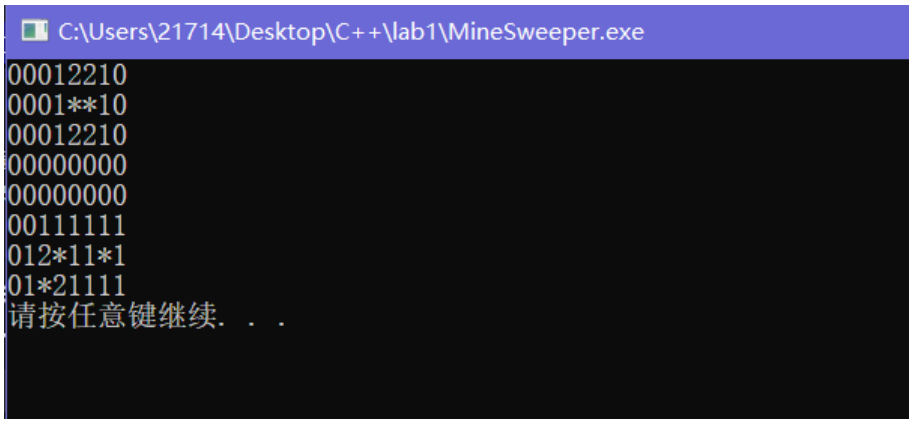


② 输入数据



③ 生成地雷图（隐蔽）

④ 通过特定操作显示地雷图（程序运行了三次的截图）



2：主要算法思路

首先定义一个类名为 MineMap，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 包含了默认构造函数，带参数的构造函数，析构函数，复制构造函数，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。（MineMap.h）

然后就是编写这些函数的代码。其中默认构造函数地雷图长、宽和地雷数量为 9,9,10。带参数的构造函数需要输入数据。同时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。（MineMap.cpp）。

最后就是组织主程序了，由于本次 lab 不要求操作，因此暂时略过。（MineSweeper.cpp）

下面详细说明地雷图生成的算法，该算法在构造函数中完成。

首先是使用 srand()函数设置随机数种子，这里种子设为(unsigned int)(std::time(NULL) + std::clock())来完成伪随机。然后利用 rand()函数随机生成横纵坐标并在地雷图上标记，将此过程循环（地雷数量）次。并且注意到多次生成坐标可能相同，故利用 count 统计不同坐标地雷的数量，利用 while 循环来控制次数，同时相同坐标则直接跳过，不增加 count 的值。

注：本次 lab 中由于已经做好了显示数字的算法，在此一同展出。因为感觉没有必要再修改程序以完成 lab 要求中非地雷位置 0 的要求，故在此说明，希望理解。

3：代码附录

① 随机数

```
std::srand((unsigned int)(std::time(NULL) + std::clock()));
int temp_x, temp_y;
int count = 0;
temp_x = std::rand() % m_length;
temp_y = std::rand() % m_width;
while (count < m_MineNum)
{
    if (m_map[temp_x][temp_y] == 0)
    {
        m_map[temp_x][temp_y] = -1;
        count++;
    }
    temp_x = std::rand() % m_length;
    temp_y = std::rand() % m_width;
}
```

② 地雷图数字

```
for (int i = 0; i < m_length; i++)
{
    int count;
    for (int j = 0; j < m_width; j++)
    {
        if (m_map[i][j] == -1)
        {
            continue;
        }
        else
        {
            count = 0;
            {
                if (i - 1 >= 0 && j - 1 >= 0 && m_map[i - 1][j - 1] == -1)
                    count++;
                if (i - 1 >= 0 && m_map[i - 1][j] == -1)
                    count++;
                if (i - 1 >= 0 && j + 1 < m_width && m_map[i - 1][j + 1] == -1)
                    count++;
                if (j - 1 >= 0 && m_map[i][j - 1] == -1)
                    count++;
                if (j + 1 < m_width && m_map[i][j + 1] == -1)
                    count++;
                if (i + 1 < m_length && j - 1 >= 0 && m_map[i + 1][j - 1] == -1)
                    count++;
                if (i + 1 < m_length && m_map[i + 1][j] == -1)
                    count++;
                if (i + 1 < m_length && j + 1 < m_width && m_map[i + 1][j + 1] == -1)
                    count++;
            }
            m_map[i][j] = count;
        }
    }
}
```

③ MineMap.h 文件

```
#ifndef MINEMAP_H_
#define MINEMAP_H_

class MineMap
{
private:
    int m_length;
    int m_width;
    int m_MineNum;
    int **m_map;
    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
```