```cpp
#include <iostream>
#include <fstream>

#include "grid.h"
// 行号列号确定vector的索引值
int grid::indexof(int row, int col) const
{
    return row * cols + col;
}
// 判断该位置是否感染
bool grid::infected(int row, int col) const
{
    return area->operator[](indexof(row, col)) == INFECTED;
}
// 未使用
bool grid::visited(int row, int col) const
{
    return true;
}
// 构造
grid::grid(string file)
{
    ifstream gridFile;
    gridFile.open(file);

    gridFile >> rows;
    gridFile >> cols;

    area = new vector<bool>(rows * cols, NOT_INFECTED);
    marks = new vector<bool>(rows * cols, NOT_INFECTED);

    while (!gridFile.eof())
    {
        int row, col;
        gridFile >> row >> col;
        area->operator[](indexof(row, col)) = INFECTED;
    }
    gridFile.close();
}
// 析构
grid::~grid()
{
    delete area;
    delete marks;
}
// 统计感染子区域大小
int grid::count(int row, int col)
{
    // 参数错误
    if (row < 0 || row >= rows || col < 0 || col >= cols)
    {
        return 0;
    }
    // 未感染
    if (grid::area->operator[](indexof(row, col)) == NOT_INFECTED)
```

```
    {
        return 0;
    }
    // 感染已标记
    if (grid::marks->operator[](indexof(row, col)) == INFECTED)
    {
        return 0;
    }
    // 感染未标记
    grid::marks->operator[](indexof(row, col)) = INFECTED;
    // 走到当前步，表明当前为感染未标记，因此加一，然后搜索周围九宫格
    return 1 + grid::count(row - 1, col - 1) + grid::count(row - 1, col) +
grid::count(row - 1, col + 1) + grid::count(row, col - 1) + grid::count(row, col
+ 1) + grid::count(row + 1, col - 1) + grid::count(row + 1, col) +
grid::count(row + 1, col + 1);
}
// 输出流
ostream& operator<<(ostream& stream, const grid& ob)
{
    for (size_t i = 0; i < ob.rows; ++i)
    {
        for (size_t j = 0; j < ob.cols; ++j)
        {
            stream << ob.area->operator[](ob.indexof(i, j));
            // 目标子区域感染+号标记
            if (ob.marks->operator[](ob.indexof(i, j)))
            {
                stream << "+";
            }
            else
            {
                stream << " ";
            }
            if (j < ob.cols - 1)
            {
                stream << " ";
            }
        }
        stream << endl;
    }
    stream << endl;
    return stream;
}
```