第二周周记

问题：

发现第一周的程序还差些判断和完善：

1.         if (num > 8 || num < 1) // 越界判断

        {

            cout << "没有这么多的拼图,请按图示控制在1~7内" << endl;

            cout << "请重新输入" << endl;

            cin >> num;

        }

已选的拼图没有判断

2.         putnotice(num);                // 生成信号

        showall(0);

每次信号显示成功后都应清除数组中的内容

3.每个区域中都有4个信号，考虑缩减信息显示密度，看起来显眼点。

2，4——5 5，1

解决：

    if (!check\_map(n))

    {

        cout << "恭喜你，你赢了！" << endl;

    }

    else

    {

        cout << "你输了，再接再厉" << endl;

    }

bool map::check\_map(int n)

{

    for (int i = 0; i < 36; i++)

    {

        switch (Level\_map[n - 1][i])

        {

        case 'A':

            if (now\_map[i] != 1 || now\_map[i] != 3)

                return false;

            break;

        case '%':

            if (now\_map[i] != 4)

                return false;

            break;

        case 'O':

            if (now\_map[i] != 2 || now\_map[i] != 0)

                return false;

            break;

        case 'X':

            if (now\_map[i] != 0)

                return false;

            break;

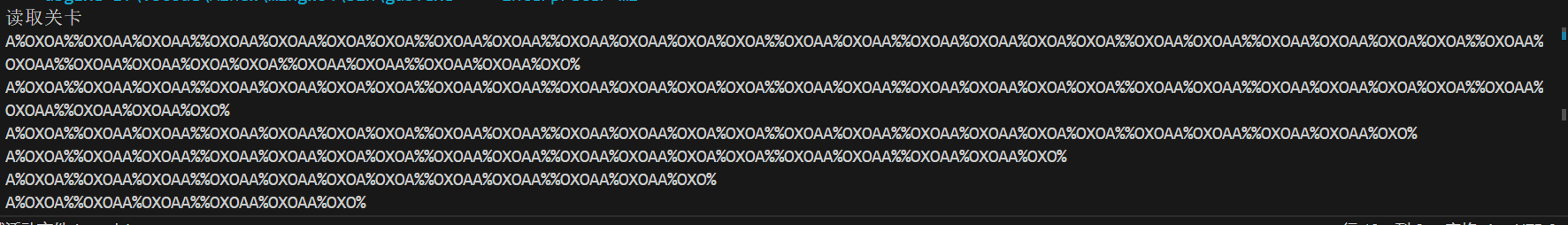
        }

    }

    return true;

}

判断逻辑，现在可以完整的进行游戏了

完成六个关卡时出现错误

结果是char的数组和getline不对付

任务：

准备完成关卡和挑战模式。

实现类Base和Hard

Base

将读取地图的功能转移到类Base中-失败，作为二维数组，地图数组无法直接返回。现在选择关卡会读取所有关卡，然后执行输入的关卡。可以修改。

    char \*empt=map.re\_map(a);

    ifstream ifs;

    ifs.open("original.txt", ios::in);

    ifs.seekg(36,ios\_base::beg);

    if (ifs.is\_open())

    {

        for (int j = 0; j < 36; j++)

            ifs >> empt[j]; // 读取每行，确保不超出数组的范围

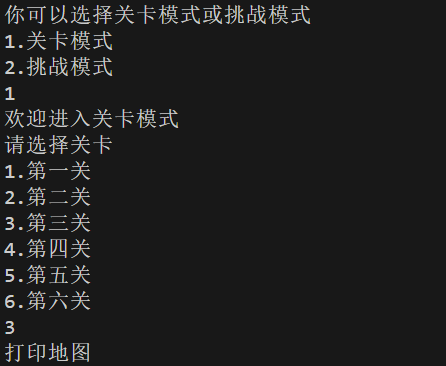
        ifs.close();

    }

    ifs.close();

    map.work(a);

利用函数可以定位输入的指针，接着用函数re\_map返回指定部分的地图数组，完成只读取一部分的任务。



Hard

随机生成地图需要一些策略

1. 对7副拼图逆运算，存储其位置数组（避免进一步特判）。然后随机其位置，结果得到的地图往往很大，或没有区块4（%）.

number = {

        {0, 1, 2, 6, 7, 8, 12, 13, 14},

        {0, 1, 2, 3, 6, 7, 8, 9, 12, 13, 14, 15, 18, 19, 20, 21},

        {0, 1, 2, 3, 6, 7, 8, 9, 12, 13, 14, 15, 18, 19, 20, 21},

        {0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 30, 31, 32, 33, 34},

        {0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 30, 31, 32, 33, 34},

        {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35},

        {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35}};

2.更改为对区块4（%）随机位置，保证至少有一个4的情况下再随机分配。即得到%的位置，再随机得到4副拼图的位置，如果能包含%，就随机其他的拼图，否则重试。

此处使用二分查找，得到能放置“4”的位置，然后根据拼图的序号选择随机范围

void hard::autocreate() // 获得随机数，得到x，y坐标，写入地图

{

    int pos4 = 0;

    int autopost[5] = {}; // 随机到了那副拼图

    int autoposi[8] = {}; // 记录哪副没有被随机,方便check

    for (;;)

    {

        pos4 = rand() % 36 + 1;

        for (;;)

        {

            for (int i = 1; i <= 4; i++)

            {

                autopost[i] = rand() % 7 + 1; // 1~7

                autoposi[autopost[i]]++;

            }

            if (recheck(autoposi))

            {

                break;

            }

        }                           // 得到一个不重复的数组

        autochoice(autopost, pos4); // 对以上四个解得到结果

        for (int i = 1; i <= 7; i++)

        {

            if (autoposi[i] != 1)

            {

                int erand = rand() % (number[i - 1].size());

                putsolution(erand, i - 1); // 得到随机数

            }

        }

        auto\_map(re\_solution());

        if (endcheck(re\_map()))

        {

            mark(re\_map());

            break;

        }

        else

        {

            for (int i = 1; i <= 7; i++)

            {

                autoposi[i] = 0;

            }

            delework();

        }

    }

}

以该函数为主体，可以加速构建合法的地图。

如上图：但是因为设计原因，当地图的值为0时，需要选择将其赋值为O或X，因此目前统一设置为O，后续如果解题困难，可以来修改

现在实现了主要的页面。

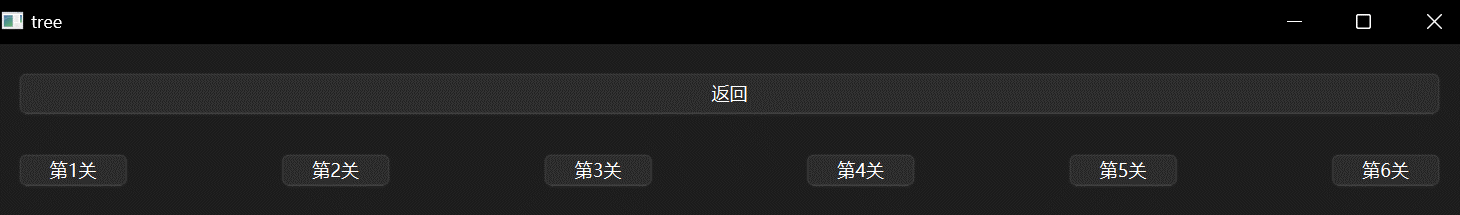
其他：

增加了提示功能，即存储解的数组，之后可以依靠这个数组给用户提示。

1. Base的该数组应从文件读取
2. Hard的需要系统给出

考虑到进度问题，qt的进度应赶上。





如上，暂时创建了一个子页面，一个主页面，实现了返回操作。

本周任务如上