一、 功能说明

这是一个魔方复原程序。由键盘输入当前魔方状态，程序将输出把魔方从当前状态转成每面颜色统一的目标状态的步骤。

具体使用方法：运行程序，输入六行字符，依次代表魔方前、后、左、右、顶、底六面的状态，每行九个字符，R 表示红，B 表示蓝，Y 表示黄，G 表示绿，O 表示橙，W 表示白。

程序将输出一串字符，由 R（面朝右面顺时针转右面）,Ri（面朝右面逆时针转右面）,D（面朝底面顺时针转底面）,Di（面朝地面逆时针转底面）,L（面朝左面顺时针转左面）,Li（面朝左面逆时针转左面）,B（面朝背面顺时针转背面）,Bi（面朝背面逆时针转背面）,F（面朝正面顺时针转正面）,Fi（面朝正面逆时针转正面）,U（面朝顶面顺时针转顶面）,Ui（面朝顶面逆时针转顶面）。按次序依次执行输出操作，即可完整还原魔方六面。

二、 设计思想

复原魔方使用的是二阶段算法（Kociemba's Algorithm, Two-Phase Approach）。

考虑魔方的表示，可以只存储 8 个角块和 12 个棱块的位置和方向。这种存储方法得到的结果和魔方本身是一一对应的。下面我们称每个面的颜色为其中心块的颜色，显然这个颜色是不会改变的，最终还原成魔方的每个面的颜色就是该颜色。对于角块的方向，选定在顶面和底面的颜色作为“参照色”，那么每个角块的方向由参照色所在的面决定。对于棱块的方向，如果这个棱块存在顶面或底面的颜色，参照色就是该颜色，否则找左面或右面的颜色作为参照色；对于顶层和底层的棱块，方向取决于参照色是否在顶面和底面；对于中间层的棱块，方向取决于参照色是否在左面和右面。

除了题目给出的 12 种旋转操作外，我们另外定义操作 R2、L2、B2、D2、F2 和 U2。其中 R2 为做两次旋转 R（或者做两次旋转 Ri，这是等价的），其他类似。

本算法的核心在于，考虑给定一个初始的魔方，如果只允许使用 U、D、L2、R2、F2 和 B2 这六个操作，那么所有角块和棱块的方向不会改变，同时中间层的四个棱块必然也在中间层。并且，所有满足这两个条件的魔方都可以仅通过这六个操作得到。根据这个思想，我们把算法分为两个阶段。在第一个阶段中，我们尝试找到一种旋转方法，把魔方变成一种状态，使得其角块和棱块的方向都是最终需要的方向，并且最终中间层的四个棱块也在中间层。在第二个阶段中，我们尝试找到一种旋转方法，把刚刚得到的魔方仅仅通过 U、D、L2、R2、F2 和 B2 这六个操作复原成最终得到的魔方。

在这两个阶段中，我们寻找旋转方法都使用的是带迭代加深的 IDA*搜索算法。具体而言，我们从小到大枚举允许使用的最大步数，然后按照深度优先搜索的顺序进行搜索，直到找到了一个合法的解。对于每个状态，可以设计估价函数，求出该状态到达目标状态所需要步数的下界，若当前已用步数加上这个下界超过了限制，就可以中止该状态的搜索。

对于估价函数的设计方法，以第一阶段的搜索为例。可以考虑仅仅针对所有角块的方向，使用广度优先搜索 BFS 预处理出每种可能的状态复原成正确的方向所需要的最小步数。那么对于一个状态，显然其复原到所需要复原的状态的步数，不会小于仅考虑其角块方向状态的步数。角块的方向最多只有3^8种状态，对

其进行预处理是可行的。

三、 自动化测试

为确保程序正确性，本组组员使用对拍进行准确性验证。对拍文件共 5 部分，已打包上传于附件中，对拍前确保其中魔方旋转程序正确性。对拍具体思路如下：通过 num1.cpp 随机数操作生成 1000 次以内的魔方旋转，将魔方初始状态（复原后）和旋转方案同时读入魔方旋转程序 num2.cpp，输出一个待复位的魔方并作为样例输入至本组魔方程序 code.cpp。旋转后将方案与待复位魔方再次读入至魔方旋转程序 num3.cpp（num3.cpp 与 num2.cpp 由于读入要求不同顾使用了两个程序），最终使用 duipai.cpp 将输出与标准魔方比对进行准确性验证。

本组共进行 500 次对拍，其中准确性得到验证，但部分测试点存在运行速度较慢情况，于是本组继续进行测试，在程序运行较慢测试点时进行中断并手动根据测试点进行代码优化，最终代码不存在超时情况。