程序设计基础大作业实验报告

一、 完成人及分工

李瑶婷 2020011852 完成 init、exchange 模块、调试程序
江楚萌 2020011847 完成 rotate、execute 模块、调试程序
洪一宁 2020011022 设计代码结构、完成 input 模块、work 模块及其子模块、测试

二、 功能说明

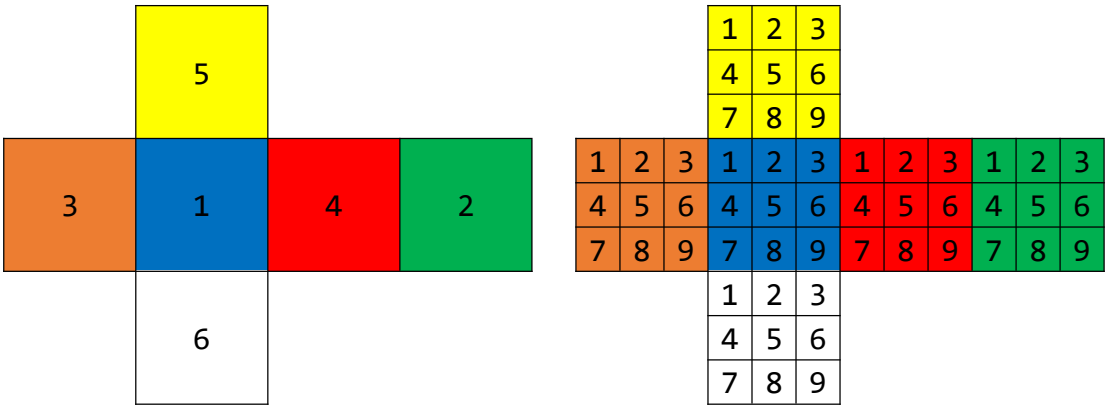
本代码接收描述一个魔方初始状态的输入，通过计算，输出操作序列，使得处于原始输入状态的魔方经过该操作序列对应的变换后可复原为每一面颜色均相同的形态。

三、 编译&运行方法

IDE 使用 Dev-C++ 5.11。
编译器使用 TDM-GCC 4.9.2 64-bit Release。

四、 设计思想

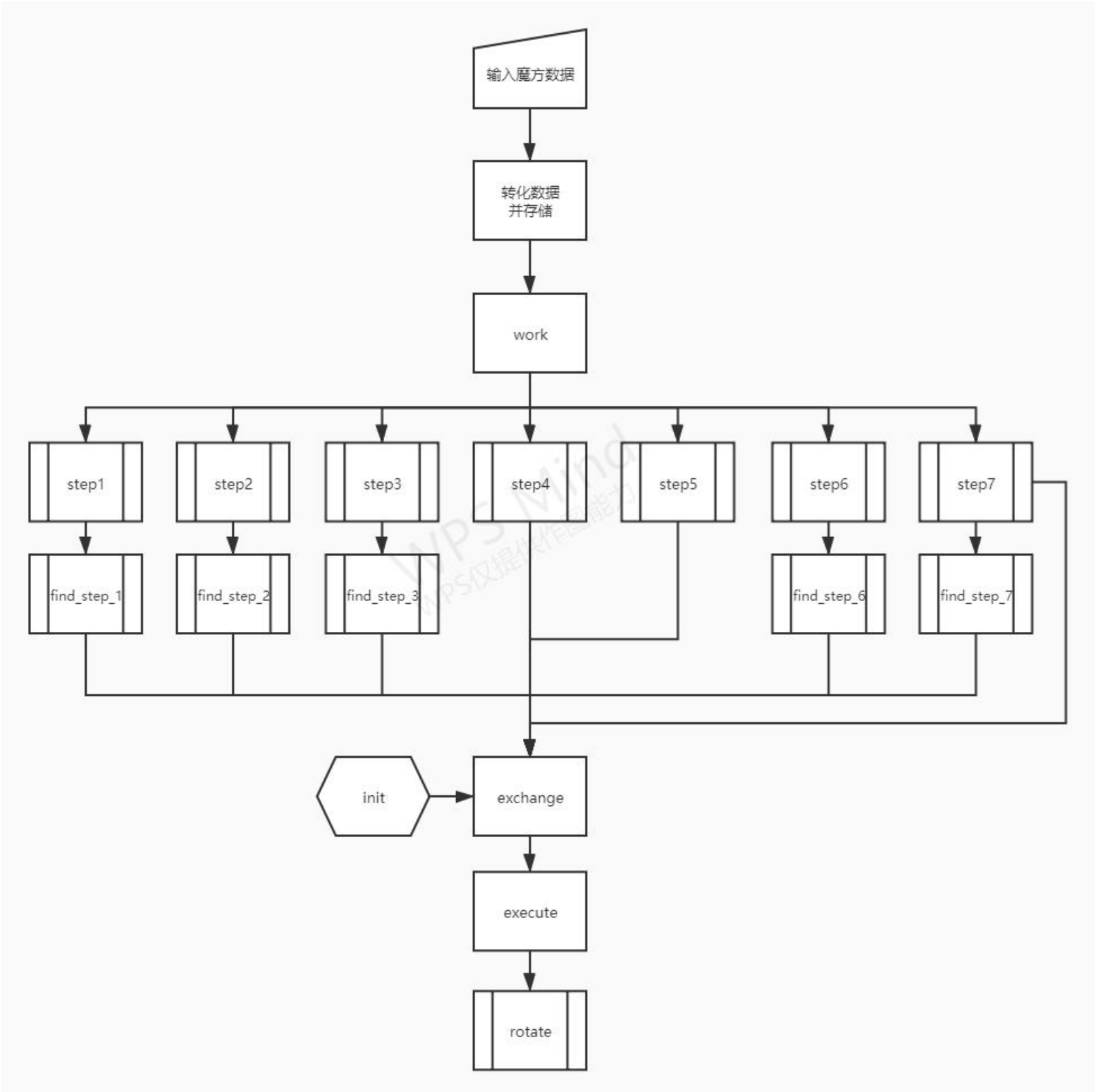
在介绍程序的设计思路前，先制定数据存储方法：
1~6 分别代表前、后、左、右、上、下面。输入的魔方存于 cube[][]数组中。数组第一维代表魔方的面，第二位代表魔方块的编号。面与块的编号方法如下表所示。表中颜色仅为区分不同面，与输入数据颜色无关。



程序主体由四部分组成：输入、决策、转化、实行。
输入（input 函数）运用 map 建立字符到数字的映射，将不易判断的字符转化为数字，从而完成数据转化。
整体看程序的主体执行部分，包含决策 work、转化 exchange 和执行 execute 三层。三个函数成嵌套关系，从顶层至底层为 work、exchange、execute

函数。其中 work 函数依据魔方复原步骤，判断当前模仿所处状态并将公式串和操作者正对的魔方面传给 exchange 函数。操作中保证正对面仅包含 1~4 面，上面必定为 5 面、下面必定为 6 面，因此正对面单个参数即可确定魔方的方向。exchange 函数将一个不确定正对面的指令串中的指令转化为正对面为 1 面的单个指令，传给 execute 函数。execute 函数接收指令后，将指令输出并对魔方进行旋转，即对 cube 数组进行更改。

流程图展示了代码中各函数及其子函数的逻辑关系。



以下分别介绍三个主要函数。

1．work 函数

work 函数将传统三阶魔方的三层复原法利用程序复现。由于对另一套公式更加熟悉，因此没有采用网络学堂下发的教程中的公式，但基本步骤相同。

第一步（step1）为复原底面棱块。

第二步（step2）为复原底面角块。

第三步（step3）为复原中层棱块。

第四步（step4）为归位顶层棱块。

第五步（step5）为归位顶层角块。

第六步（step6）为调整顶层角块顺序。

第七部（step7）为调整顶层棱块顺序。

考虑到一个人在复原魔方时，进行每一步时会首先考虑当前魔方是否已经完成了这一步的复原目标。如果是，则进入到下一步；如果否，则任意寻找一个尚未归位的块对其进行相应操作（step1、2、3）或按公式判定规则复原（step4、5、6、7）。由于存在判定操作，因此运用 while 循环，判定通过则跳出，否则进行操作即可。为理清逻辑，部分操作较为复杂的步骤如 step1、2、3、6、7 使用了子函数 find_step_1、2、3、6、7 实现。

2．exchange 函数

exchange 模块是为了简化 execute 模块而设计的，主要功能是将面对编号 1-4 面的 48 条旋转指令全部转化为面对编号为 1 面的 12 条旋转指令，然后调用 execute 完成指令。此处的简化指令思想和 execute 中将逆时针指令转化为顺时针从而将指令数进一步减少到 6 的思想相同，可以大大简化实际操作的旋转程序，使执行层逻辑清晰。

实现方法则采用了“打表”法（init 函数），穷举所有可能指令并进行转化，比较暴力啦（⊙__⊙；）。

3．execute 函数

1）输出指令

根据任务，按照格式输出转化指令即可。

2）指令转化

输入的指令一共有 12 个，首先将之转化为 6 个，以便之后进一步的操作。

实现方法为将逆时针的指令转化为顺时针旋转 3 次（小写指令转为大写，并使旋转次数 num 为 3）。

3）旋转魔方

观察到每一个指令下，魔方均有四面有 3 个块会改变，一个面顺时针旋转 8 个块改变，因此在每个指令下用循环完成 3 个块改变的四个面的旋转，建立 rotate 函数完成 8 个块改变的一个面的顺时针旋转。

①四面旋转：在 cube 二维数组中用公式计算出每个指令对应的结果；

②一面旋转（rotate 函数）：将该面 9 个块的颜色赋给一个二维数组（temp0），旋转完成之后赋回 cube 数组。

五、 实验感想与收获（可选）

李瑶婷：

超级感谢同组负责决策层的一宁~其实在做大作业之前本人并不会玩魔方，开始时也觉得反正程序可以解决，只要执行公式就好了，但是写的时候就真的无从下手。后来大家聚在一起开会，信竞加魔方竞速大佬一宁清晰的思路真的让我学到了很多，回去就顺便学会了魔方。虽然是我和楚萌负责执行层，但一宁也会给出超有建设性的建议！

体会最深的就是层层简化指令的过程，我们不应该执行到哪才想到哪写到哪，而应在最开始的时候先有整体的把握和预见，然后在源头上尽可能降低后续的复杂度。这就是后来由我去写转化层的原因，而 exchange 和 init 函数的逻辑与操作是非常简单而集中的，在 debug 过程中也省了不少力。

江楚萌：

开学听到大作业是旋转魔方的时候，觉得这实在是太酷了，但是简直是无从下手。可是，经过大半个学期的学习，我们其实也可以通过自己的努力实现这样一个好玩的小程序。在写作业的过程中切实体会到了很多课上提到过的东西，也跟同组的大佬们学会了很多东西，真正体会到了编程是一门实操的课程。

要完成这样一个比较大的任务，一口吃是吃不掉的，首先的任务的就是将它分块，不管该怎么实现，但是先把要写什么函数分出来，不一“main”到底，或许分出函数其实也就是编程最基本的分工。在这之后，大家各自写好自己的代码，再拼接之后，就可以完成一个人很难完成的工作了。

在实现自己这块内容时，体会最深的减少代码的重复，顺时针旋转其中要改变 8 个块的那一面时，对于不同指令的代码是完全相同的，因此可以提出来构建一个统一的函数，提醒自己以后在写代码的时候也要学会去发现不同步骤中的相同之处减少重复。

洪一宁：

在写魔方大作业的过程中有很多收获，其中最让我印象深刻的还是在写的过程中的效率问题。解决较复杂问题的时候很容易思维混乱，尤其是在精力并不充沛的时候，写出来的东西常常要改很多。但如果多分几个子函数，一小段一小段解决问题，正确率与效率都会高很多，也增加了代码的可读性。

此外，借助工具完成任务也很有用。刚开始写代码的时候，曾经试图不借助实体魔方写程序，随后发现这太难了。但拿到魔方之后再写就容易了很多。一个实体魔方对于程序的调试也很有用。

六、 其他（可选）

在建构程序之外，我们对程序进行了测试。测试程序与文件位于附件中“for test”文件夹中。

其中，checker.exe 为测试总程序，负责协调各程序运行先后顺序及输入输出文件。程序中常数 t 为测试组数，为 1000。

在每轮测试中，checker.exe 将首先输出当前测试为第几组，随后运行 op.exe，op.exe 将随机生成一段长度为 1~1000 的操作序列输入至 op.in 中。data.exe 读取 op.in 中的操作序列并依此对一个已复原魔方进行操作。操作使用的函数 rotate 与 code.cpp 中完全相同。由于生成数据的操作与复原魔方的

操作没有直接联系，即操作必定不相同，也必定不互为逆操作，因此重复利用 rotate 函数测试是安全的。随后 data.exe 将操作得到的魔方依据原题目输入格式输出至 data.in 中，作为输入文件。这一输入文件生成方法保证了魔方的合法性，即魔方必定有解。至此，数据生成部分结束。

随后，checker.exe 将 data.in 输入至 code.exe，即待检查的程序中。code.exe 经过计算将运算结果，即操作序列输出至 cube.out。至此，运行部分结束。

最后，checker.exe 将 cube.out 中的操作序列输入至 rotate.exe 中。rotate.exe 通过文件读写读取原始输入数据，即 data.in 中的魔方形态，并通过另一种不同于 cube.cpp 中 rotate 函数的实现方法依据序列对魔方进行操作，并将操作得到的魔方输出至 end.out 中。随后，checker.exe 将 end.out 输入至 see.exe 中，see.exe 将检查魔方是否处于复原状态，如是则向 ed.out 输出“1\n”，否则向 ed.out 输出“-1\n”。随后，checker.exe 通过比较指令比较 ed.out 与 std.out 是否相同；std.out 为纯文本文件，文本内容为 1 以及随后的换行。如果两文件相同，那么可以确定程序运行结果正确，完成该轮测试；如果两文件不同，则 checker.exe 停止运行，可利用仍存于 data.in 中的数据调试。

经过测试，我们发现已经通过 OJ 小样例测试的程序仍存在错误。改正后，程序通过了 1000 组测试。