同济大学

实验报告

高程大作业 Hanoi

姓名：雷翔

学号：2053932

完成日期：2022.11

1. 题目

1.1 题目简介

题目总体分成两个部分：1. 整合之前做的有关汉诺塔的题目；2. 添加图形化演示功能。

1.2 题目分析

题目要求实现一个菜单，让用户通过菜单进行选择，从而执行相应的选项。选项 1、2、3、4 主要是之前做过的题目，在这里需要做一个整合；选项 5、6、7、8 则层层递进，一步一步添加图形化演示功能；选项 9 是汉诺塔的游戏版，通过用户手动输入来实现盘子的移动。

1.3 题目具体要求

1.3.1 选项 1、2、3、4 要求

主要是输出汉诺塔的移动步骤。选项 1、2、3 都是选项 4 的子集：选项 1 仅要求输出基本解；选项 2 要求输出带步数记录的基本解；选项 3 则是要求在选项 2 的基础上再加上输出横向数组；选项 4 则是要求在选项 3 的基础上再加上输出纵向数组。

1.3.2 选项 5、6、7、8 要求

主要是画出汉诺塔的图形化界面及移动过程。选项 5、6、7、8 层层递进：选项 5 要求画出三根圆柱；选项 6 要求在起始圆珠上画出 n 个圆盘；选项 7 要求完成起始柱最上层圆盘的移动；选项 8 要求完成汉诺塔移动的整个过程。选项 7、8 中圆盘的移动要求先上移、再平移、再下移。

1.3.3 选项 9 要求

将游戏模式由自动实现更改成人工操作。要求检查输出的合理性，不符合移动规则（如大盘压小盘、圆柱为空等）要输出错误信息；每次合理的移动要记录下步数。当所有盘子从起始柱移动到目标柱后，游戏结束，输出提示信息。

2. 整体设计思路

2.1 题目分析

题目的关键：改变三个数组的值和对应的数组指针，数组里的值表示柱子上的盘子，数组指针表示柱子上盘子的个数，选项 1、2、3、4、8 均通过同一个递归函数实现。数组值和数组指针的改变本质是一个数（也可以成一个盘子）从起始柱对应的数组移动到目标柱对应的数组，其结果是起始柱盘子数减 1，目标柱盘子数加 1。针对选项 4，需要在屏幕上输出横向数组和纵向数组；针对选项 8，需要在伪图像化界面上将盘子从起始柱移动到目标柱；针对选项 9，需要对用户手动输入的操作进行判断，如是否可以移动，若不可以移动，要提示用户，若可以移动，进行相关操作，移动完成后，还需判断游戏是否结束。

以上就是题目的整体设计思路，下面给出项目涉及到的主要操作函数。

```
1.    void to_be_continued(const int X, const int Y)
2.  一轮游戏结束后光标移动
3.   int Menu()
4.  显示菜单，并返回用户正确选项
5.   void hanoi(int n, char src, char tmp, char dst, int choice)
6.  递归函数
7.   void DoByChoice(int n, char src, char tmp, char dst, int choice)
8.  根据用户选项来决定执行何种操作 选项：1、2、3、4、8
9.   void DoByChoice2(int n, char src, char tmp, char dst, int choice)
10. 根据用户选项来决定执行何种操作 选项：6、7
11.  void DoByChoice3(int n, char src, char dst)
12. 根据用户选项来决定执行何种操作 选项：9
13.  void GuiLing()
14. 每次执行完一次用户选项后，静态全局变量置原值
15.  int IsMove(char from_ch, char to_ch)
16. 选项 9 判断要不要移动 返回 0 表示会大盘压小盘，返回-1 表示起始柱为空，返回大于 0 表示要移动盘
    的编号
17.  void MovePanInArray(int n, char src, char dst)
18. 将盘子从 src 对应的数组移动到 dst 对应的数组
19.  void PrintByChoice(int n, char src, char tmp, char dst, int choice);
20. 紧接上一个函数，在屏幕上显示相应内容 选项 1、2、3、4、8
21.  void MovePanFromSrcToDst(int id, char src, char dst)
22. 在图形化界面中，将 src 对应柱子的最上面的盘子移动到 dst 对应柱子上
23. 先移动数组中元素，再移动图形化界面！！！
24.  void GoToHigh(int one_x, int one_y, int another_x, int another_y, int id, int len)

25. 盘子上移
26.  void GoToInLine(int one_x, int one_y, int another_x, int another_y, int id, int le
    n)
27. 盘子平移
28.  void GoToLow(int one_x, int one_y, int another_x, int another_y, int id, int len)

29. 盘子下移
30.  void InputSrcDst(int* n_pre, char* src_ptr, char* tmp_ptr, char* dst_ptr)
31. 输入盘子个数、起始柱和目标柱
32.  void InputMoveSpeed()
33. 输入移动速度
34.  void InitSrcArray(int n, char src)
35. 初始化：往起始柱填盘子编号；越大的盘子，编号越大
36.  void InitForOp4(int n, char src, char dst)
```

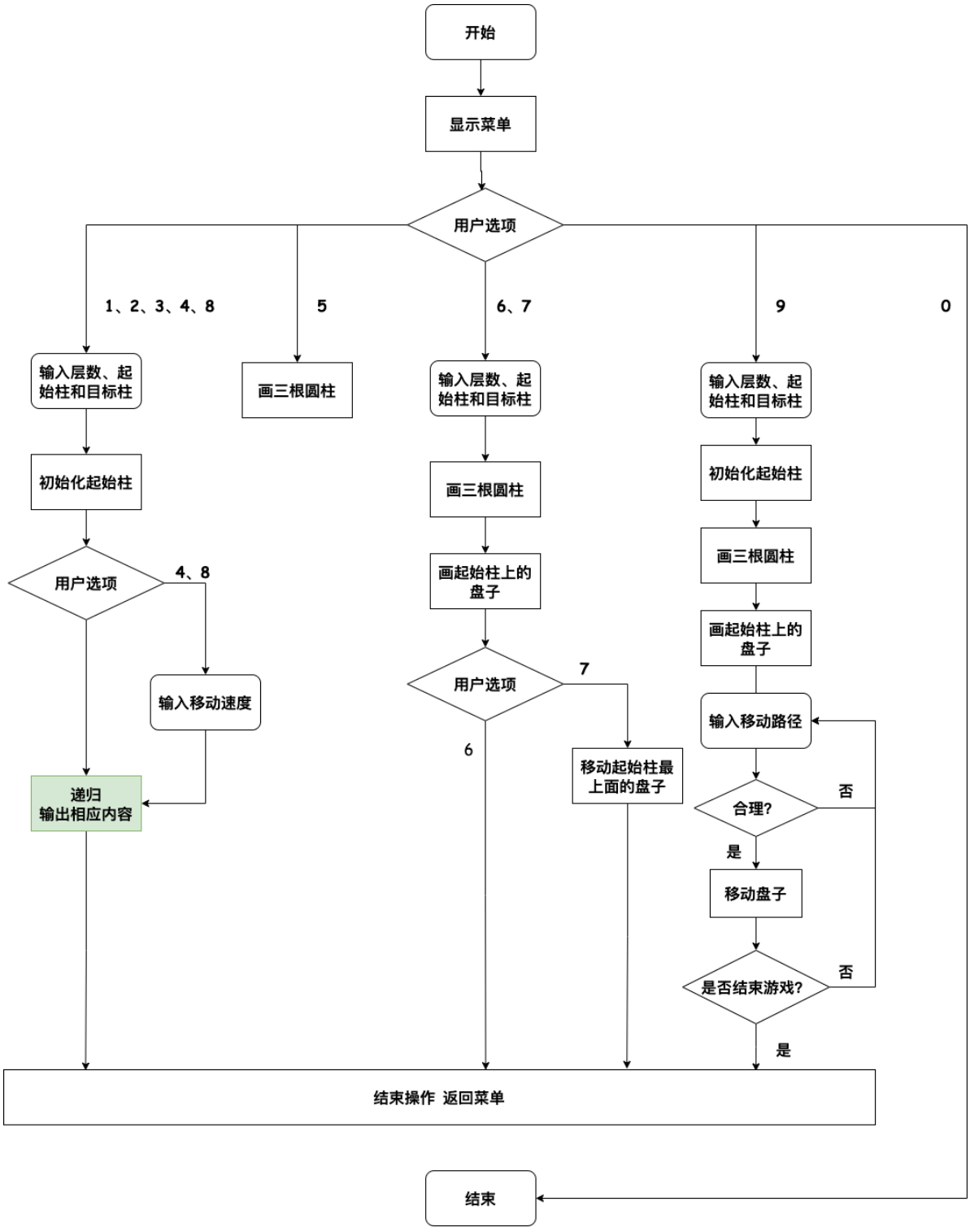
```
37. 输出选项 4 的界面：第一行显示内容、纵向打印、横向打印
38.  void InitForOp8(int n, char src, char dst)
39. 输出选项 8 的界面：第一行显示内容、纵向打印、横向打印 + 圆柱 + 起始柱
40.  void DrawTriYuanZhu()
41. 画三根圆柱
42.  void DrawSrcPanZi(int n, char src)
43. 画初始化时起始柱上的盘子
44.  void PrintInX()
45. 打印横向数组
46.  void PrintInY(int offset_x, int offset_y)
47. 打印纵向数组 不包括底座(A.B.C 和一排等号)
48.  void Delay(int x)
49. 延时设置
```

2.2 关于 arrayX 和 topX 的几点说明

- 三个一位数组 arrayA、arrayB、arrayC 存储圆柱上的盘子，元素值为 1～10，同时 1～10 也代表着盘子的大小。
- 值为 1 的盘子最小，值越大，盘子越大
- 三个数组指针 topA、topB、topC 分别表示对应数组中盘子的个数
- topA+topB+topC 之和始终等于 n
- topX=n：表示盘子全在 X 柱上；topX=0：表示该柱没有盘子
- 在选项 9 中，若目标柱对应数组的元素个数等于 n，说明游戏结束

3. 主要功能的实现

3.1 功能实现流程图



3.2 菜单界面 Menu()函数

首先在屏幕上显示菜单，然后让用户进行选择，在函数内部对用户的选择进行判断，如果输入不是0～9的数字，输入无效，并且此时屏幕上不会有任何显示，直到用户输入0～9之间的数字，返回该值。注意函数返回值是int型。

3.3 输入函数

1. InputSrcDst(int *n_ptr, char *src_ptr, char *tmp_ptr, char *dst_ptr)
2. 函数接受4个参数，通过以指针作为形参，从而实现从函数中输入变量值也能改变主函数变量值的目的。
3. InputMoveSpeed()
4. 移动速度move_speed为静态全局变量，因此不需要给函数传参，同样也在本文件内改变变量值。

3.4 初始化起始柱 InitSrcArray(int n, char src)

首先需要确定起始柱是A、B、C柱中的哪一个，然后再依次向其对应的数组中从下标0开始存入盘子的编号，编号从大到小开始存。盘子越大，编号越大；盘子越小，编号越小。

3.5 输出数组

1. PrintInX()：输出横向数组
2. PrintInY(int offset_x, int offset_y)：输出纵向数组，两个参数是输出偏移量，位置基准由const常量给出。

3.6 根据选项执行

1. DoByChoice(int n, char src, char tmp, char dst, int choice)：调用递归函数，输出相应的内容
2. DoByChoice2(int n, char src, char tmp, char dst, int choice)：选项6和选项7只有最后面不同，所以封装成一个函数
3. DoByChoice3(int n, char src, char dst)：避免main文件中选项9实现代码过长，所以封装起来

以上三个函数的具体实现过程见6.3。

3.7 移动盘子

```
1. void MovePanFromSrcToDst(int id, char src, char dst);
2. void GoToHigh(int one_x, int one_y, int another_x, int another_y, int id, int len);
       // 上移
3. void GoToInLine(int one_x, int one_y, int another_x, int another_y, int id, int len
   );  // 平移
4. void GoToLow(int one_x, int one_y, int another_x, int another_y, int id, int len);
       // 下移
```

将盘子移动过程：先上移、再平移、再下移，分成三个函数来实现。

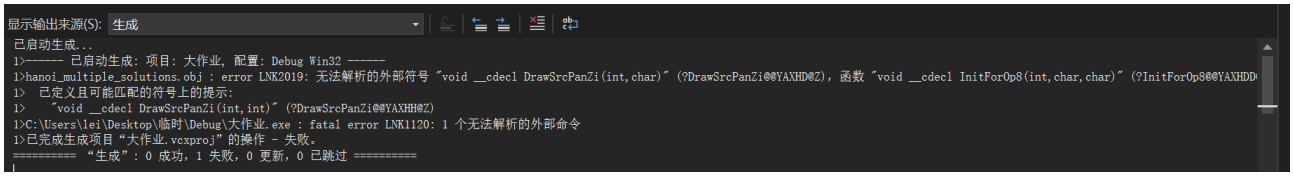
4. 调试过程碰到的问题

4.1 每次执行完一个操作后没有重置静态全局变量

执行完一次操作，用户没有退出菜单，而是选择执行下一次操作时，界面显示结果不对，最明显就是步数不是从 1 开始。经过调试，发现是这两次操作的静态全局变量的值累加在一起，因此，我封装了一个函数 GuiLing，每次执行完选项对应的所有操作后调用这个函数就好了，问题得以解决。

4.2 链接错误 error LNK2019

在重写代码的过程中，出现了一个 link 错误。具体原因：DrawSrcPanZi(int n, char src) 在函数定义时把 src 的类型写成了 int。这种错误很难发现，事实证明，要仔细看错误信息。



5. 完成本次作业心得体会

5.1 完成本次作业得到的一些心得体会，经验教训

1. 当程序遇到 bug 时，在合适的位置输出 cout 一串字符来定位，看是否符合自己的预期，如果符合预期，说明该处没有错误，换下一个位置输出一串字符，直到找到错误出现的位置。
2. 一步一步完成程序，写好一个功能点时要及时验证，而不是全写完才验证，这样可以缩小错误出现的范围。如果需要奇怪的 bug，死活找不到时，果断一点，直接删除写好的代码，重写这个功能。
3. 善于利用 vs 自带的调试功能，有时错误可能是因为逻辑处理没写好引起的，这时候调试功能就非常有用，可以帮助自己知道编译器执行程序的流程，看是否与自己想的流程相同，若不同，说明逻辑有问题。
4. 如果必须使用静态全局变量，要格外注意！
5. 函数和变量的命名规范：函数命名遵循大驼峰原则，局部变量采用下划线命名法，之所以使用 topA，而不是用 top_a，主要是因为我觉得前者更美观。总体来说，变量名要“名副其实”，让使用者在之后通过名称就能看出变量的作用范围或目的。
6. 多写函数，尽量将一个函数的代码量控制在 50 行以内。
7. 少用数字，多用 const 变量，可以以基准+偏移量的方式来定位。

5.2 在做一些较为复杂的程序时，是分为若干小题还是直接一道大题的形式更好

我觉得分成若干小题的形式更好，通过前几个小题的铺垫，我们可以找到题目在数学上的实现过程，更加专注于代码的具体实现。

5.3 汉诺塔完成了若干小题，总结你在完成过程中是否考虑了前后小题的关联关系，是否能尽可能做到后面小题有效利用前面小题已完成的代码，如何才能更好地重用代码

本题的前四个小题都是前面我们做过的题目，题目之前是层层递进的，尤其是第 4 题，集合了前 3 题的输出方式：横向+纵向，当初做这道题目的时候就将其封装成了两个函数，不过在做本题时，我觉得原来的函数不够清新，更换成了 PrintInX 和 PrintInY，更容易通过函数名来理解函数的作用。

第 5 题到第 8 题也是一样的，写后一题时也会使用前一题封装的函数。尽可能多的调用函数，来让程序的逻辑更加清晰。

为了更好地重用代码，每当我们写一个函数时，应该及时写清注释，方便之后再次调用。随着项目的逐渐复杂，我们应该更加关注函数的作用，而不是函数的具体实现，因此，注释非常重要。

5.4 已本次作业为例，说明如何才能更好地利用函数来编写复杂的程序

首先第一点，要仔细阅读项目要求文档。我自己是第一次接触这种规模的作业，单个问题解决起来其实并不复杂，作业难点主要在于如何设计函数，来实现更好的复用性。

在本次作业中，我的做法是，先从前往后把程序给写出来，把功能点封装成函数，写好函数的注释，一步一步测试，直到作业完成。然后总览整个源程序，把用文字详细记录操作流程，在这份记录中，我修改了部分函数的名字，参数，还将一个大函数拆分成两个小函数等等。然后根据这个流程，重开一个项目，大部分代码从原代码中直接拷贝过来，需要修改的地方就修改。我认为这样写出来的代码会比第一次写出来的要好很多。

- Menu()
- 根据用户输入选项执行对应的操作
  - 选项1、2、3、4、8
    - 需要输入层数、起始柱和目标柱 封装一个函数

      `InputSrcDst(int *n_ptr, char *src_ptr, char *tmp_ptr, char *dst_ptr)`利用指针作为函数参数的方式，得到src、tmp和dst的值
    - 下面内容封装成一个统一函数 `DoByChoice(int n, int src, int tmp, int dst, int choice)`
    - 初始化起始柱 主要是把n个盘子所对应的数字依次放到起始柱对应的数组中

      `InitSrcArray(char src)`
    - 对选项4和选项8，还需要输入移动速度

      `InputMoveSpeed()` 给静态全局变量move_speed赋值，注意静态局部变量有默认值0
      - 清屏
      - 选项4和选项8的页面不同，要分开输入 -> 拆成两个函数 `InitForOp4` `InitForOp8`
    - 调用递归函数 `hanio`
      - 在递归函数中移动盘子以及根据用户选项输入从而进行相应操作封装成两个函数 重新写一下

        `MovePanInArray(char src, char tmp, char dst)`

        `PrintByChoice(int n, char src, chat tmp, char dst, int choice)`
      - 重点是第二个函数
        - 选项1、2、3、4都是以前的，主要是选项8，下面重点说选项8
        - 先在三个数组中移动盘子，用函数 `MovePanInArray` 实现，再在图形柱子上移动盘子，写一个函数 `MovePanFromSrcToDst(id, src, dst)` id：盘子的标号，避免在移动过程中盘子的颜色发生变化
      - 选项4和选项8在屏幕上显示横向数组和纵向数组的位置也不一样
        - 显示横向数组好解决，在cout xxx 后接着数组就可以 `PrintInX()`
        - 显示纵向数组通过函数传参来解决 基准 + 偏移量 基准可以定义成const常量 `PrintInY(int offset_x, int offset_y)`
  - 选项5
    - 清屏
    - 画出三根圆柱 `DrawTriYuanZhu();`
  - 选项6
    - 同选项1、2、3、4、8的步骤一：需要输入层数、起始柱和目标柱 封装一个函数
    - 清屏
    - 在终端第一行输出 "从src移动到dst，共n层"
    - 画出三根圆柱 `DrawTriYuanZhu();`
    - 画出起始柱上的所有盘子 `DrawSrcPanZi(char src)`
  - 选项7
    - 同选项1、2、3、4、8的步骤一：需要输入层数、起始柱和目标柱 封装一个函数
    - 清屏
    - 在终端第一行输出 "从src移动到dst，共n层"
    - 画出三根圆柱 `DrawTriYuanZhu();`
    - 画出起始柱上的所有盘子 `DrawTriPanZi(char src)`
    - 移动起始柱最上面的盘子

    可以看到选项6和选项7只有最后一步有差别，所以可以将它们放在一起

6. 附件：源程序

源程序一共分成四部分，分别是：主函数、菜单函数、几个操作函数和头文件。

6.1 主函数

```
1.  int main()
2.  {
3.      /* demo 中首先执行此句，将 cmd 窗口设置为 40 行 x120 列（缓冲区宽度 120 列，行数 9000 行，即 cmd 窗口右侧带有垂直滚动杆）*/
4.      cct_setconsoleborder(120, 40, 120, 9000);
5.
6.      int n;  // 盘子个数
7.      char src, tmp, dst;
8.      while (1)
9.      {
10.         int user_choice = Menu();
11.         int tag = 1;  // tag = 0 则退出死循环
12.         switch (user_choice)
13.         {
14.             case 1:
15.             case 2:
16.             case 3:
17.             case 4:
18.             case 8:
19.                 InputSrcDst(&n, &src, &tmp, &dst);  // 完成盘子个数、起始柱、中间柱和目标柱的输入
20.                 DoByChoice(n, src, tmp, dst, user_choice);
21.                 break;
22.             case 5:
23.                 cct_cls();  // 清屏
24.                 DrawTriYuanZhu();
25.                 break;
26.             case 6:
27.             case 7:
28.                 InputSrcDst(&n, &src, &tmp, &dst);  // 完成盘子个数、起始柱、中间柱和目标柱的输入
29.                 DoByChoice2(n, src, tmp, dst, user_choice);
30.                 break;
31.             case 9:
32.                 InputSrcDst(&n, &src, &tmp, &dst);  // 完成盘子个数、起始柱、中间柱和目标柱的输入
33.                 DoByChoice3(n, src, dst);
34.                 break;
35.             case 0:
36.                 tag = 0;
37.                 break;
38.         }
39.         if (tag == 0)
40.             break;
41.         to_be_continued(0, 38);
42.     }
43.
44.     return 0;
45. }
```

6.2 菜单函数

```
int Menu()
{
    cout << "-------------------------------- " << endl;
    cout << "1.基本解" << endl;
    cout << "2.基本解(步数记录)" << endl;
    cout << "3.内部数组显示(横向)" << endl;
    cout << "4.内部数组显示(纵向 + 横向)" << endl;
    cout << "5.图形解 - 预备 - 画三个圆柱" << endl;
    cout << "6.图形解 - 预备 - 在起始柱上画 n 个盘子" << endl;
    cout << "7.图形解 - 预备 - 第一次移动" << endl;
    cout << "8.图形解 - 自动移动版本" << endl;
    cout << "9.图形解 - 游戏版" << endl;
    cout << "0.退出" << endl;
    cout << "-------------------------------- " << endl;
    char user_choice;
    cout << "[请选择:] ";
    while (1)
    {
        user_choice = _getch();  // 读取单个字符 （无缓冲区 无回显）
        if (user_choice >= '0' && user_choice <= '9')
        {
            cout << user_choice << endl;  // 正确输入时显示用户选项
            break;
        }
    }
    cout << endl;
    return user_choice - '0';  // 返回的是整型
}
```

6.3 几个操作函数

有删减，只列举了几个比较重要的函数，具体内容见代码文件。

```
void hanoi(int n, char src, char tmp, char dst, int choice)
{
    if (n == 0)
        return;
    hanoi(n - 1, src, dst, tmp, choice);
    pace_count++;
    MovePanInArray(n, src, dst);  // 将盘子从 src 对应的数组移动到 dst 对应的数组
    PrintByChoice(n, src, tmp, dst, choice);  // 根据用户选项来显示对应结果
    hanoi(n - 1, tmp, src, dst, choice);
}
void DoByChoice(int n, char src, char tmp, char dst, int choice)
{
    InitSrcArray(n, src);
    if (choice == 4)
    {
        InputMoveSpeed();  // 输入移动速度
        cct_cls();  // 清屏
        InitForOp4(n, src, dst);  // 显示初始化界面
        Delay(move_speed);
    }
    else if (choice == 8)
    {
```

```
23.         InputMoveSpeed();  // 输入移动速度
24.         cct_cls();  // 清屏
25.         InitForOp8(n, src, dst);  // 显示初始化界面
26.         Delay(move_speed);
27.     }
28.     hanoi(n, src, tmp, dst, choice);
29.     GuiLing();
30. }
31. void DoByChoice2(int n, char src, char tmp, char dst, int choice)
32. {
33.     InitSrcArray(n, src);
34.     cct_cls();  // 清屏
35.     cct_setcursor(3);  // 不显示光标
36.     cout << "从 " << src << " 移动到 " << dst << "，共 " << n << " 层" << endl;
37.
38.     DrawTriYuanZhu();  // 初始状态
39.     DrawSrcPanZi(n, src);
40.
41.     if (choice == 7)
42.     {
43.         Sleep(1000);  // 延迟 1 秒
44.
45.         if (n % 2 != 0) // 奇数个盘子，第一个从起始柱向目标柱移动
46.         {
47.             MovePanInArray(n, src, dst);
48.             MovePanFromSrcToDst(1, src, dst);  // 移动图形盘子  第一个参数给 1 只移动
    最上面的 颜色固定
49.         }
50.         else            // 偶数个盘子，第一个从起始柱向中间柱移动
51.         {
52.             MovePanInArray(n, src, tmp);
53.             MovePanFromSrcToDst(1, src, tmp);
54.         }
55.     }
56.     GuiLing();
57. }
58. void DoByChoice3(int n, char src, char dst)
59. {
60.     InitSrcArray(n, src);
61.     cct_cls();  // 清屏
62.     cct_setcursor(3);  // 不显示光标
63.     InitForOp8(n, src, dst);  // 显示初始化界面
64.
65.     cct_gotoxy(0, 18 + OffsetY);
66.     cout << "请输入移动的柱号(命令形式：AC=A 顶端的盘子移动到 C，Q=退出) : ";
67.     int pos_x, pos_y; // 当前光标所在位置
68.     cct_getxy(pos_x, pos_y);
69.     while (1)
70.     {
71.         cct_gotoxy(pos_x, pos_y);
72.         char str[20] = { '\0' };  // 存放输入字符
73.         int count = 0;  // 记录输入字符的个数
74.         while (count < 20)  // 这里的逻辑比较麻烦，主要是为了跟老师 demo 一样
75.         {
76.             char ch = _getch();  // 读取回车返回 \r 13
77.             if (ch != ' ')
78.                 cout << ch;
79.             else
80.                 continue;
```

```
81.             if (ch >= 'a' && ch <= 'z')
82.                 ch -= 'a' - 'A';
83.             str[count] = ch;
84.             count++;
85.             if (str[0] == 'Q' && str[1] == '\r' || str[0] == 'q' && str[1] == '\r')

86.             {
87.                 cout << endl << "游戏中止!!!!!" << endl;
88.                 return;
89.             }
90.             // str[0]表示出发柱 str[1]表示到达柱，即从 str[0]到 str[1]
91.             if (count == 3 && str[0] >= 'A' && str[0] <= 'C' && str[1] >= 'A' && st
    r[1] <= 'C' && str[0] != str[1] && str[2] == '\r')
92.                 break;
93.             if (str[count - 1] == '\r' || count == 19)
94.             {
95.                 // 最多只能输 19 个字符，多了就清
96.                 cct_gotoxy(pos_x, pos_y);
97.                 cct_showch(pos_x, pos_y, ' ', COLOR_BLACK, COLOR_WHITE, 20);
98.                 cct_gotoxy(pos_x, pos_y);
99.                 count = 0;
100.            }
101.        }
102.        int id = IsMove(str[0], str[1]);
103.        if (id == -1)
104.        {
105.            cct_gotoxy(0, pos_y + 1);
106.            cout << "源柱为空!";
107.            Sleep(1000);
108.            cct_showch(pos_x, pos_y, ' ', COLOR_BLACK, COLOR_WHITE, 20);
109.            cct_showch(0, pos_y + 1, ' ', COLOR_BLACK, COLOR_WHITE, 40);
110.        }
111.        else if (id == 0)
112.        {
113.            cct_gotoxy(0, pos_y + 1);
114.            cout << "大盘压小盘，非法移动!";
115.            Sleep(1000);
116.            cct_showch(pos_x, pos_y, ' ', COLOR_BLACK, COLOR_WHITE, 20);
117.            cct_showch(0, pos_y + 1, ' ', COLOR_BLACK, COLOR_WHITE, 40);
118.        }
119.        else
120.        {
121.            pace_count++;
122.            MovePanInArray(id, str[0], str[1]);  // 原本下一行在这一行上面，上传版
    没修改 （未验证）
123.            cct_gotoxy(0, 17 + OffsetY);
124.            cout << "第" << setw(4) << pace_count << " 步
    " << "(" << setw(2) << id << "#: " << src << "-->" << dst << ")";
125.            PrintInX();
126.            PrintInY(OffsetX, OffsetY);
127.            MovePanFromSrcToDst(id, str[0], str[1]);
128.            cct_showch(pos_x, pos_y, ' ', COLOR_BLACK, COLOR_WHITE, 20);
129.        }
130.
131.        // 判断游戏是否结束
132.        if (dst == 'A')
133.        {
134.            if (n == topA)
135.            {
136.                cct_gotoxy(0, pos_y + 1);
```

```
137.                     cout << "游戏结束!!!!!" << endl;
138.                     break;
139.                 }
140.             }
141.             else if (dst == 'B')
142.             {
143.                 if (n == topB)
144.                 {
145.                     cct_gotoxy(0, pos_y + 1);
146.                     cout << "游戏结束!!!!!" << endl;
147.                     break;
148.                 }
149.             }
150.             else
151.             {
152.                 if (n == topC)
153.                 {
154.                     cct_gotoxy(0, pos_y + 1);
155.                     cout << "游戏结束!!!!!" << endl;
156.                     break;
157.                 }
158.             }
159.         }
160.         GuiLing();
161.     }
162.     void MovePanFromSrcToDst(int id, char src, char dst)
163.     {
164.         int src_x, src_y, dst_x, dst_y;
165.         int cur_pan_len;  // 当前移动的盘子的长度
166.         // 确定移动的起点和终点的位置
167.         if (dst == 'A')
168.         {
169.             if (src == 'B')
170.             {
171.                 cur_pan_len = arrayA[topA - 1] * 2 + 1;
172.                 src_x = 11 - arrayA[topA - 1] + 33;
173.                 src_y = 14 - topB;
174.                 dst_x = src_x - 33;
175.                 dst_y = 15 - topA;
176.             }
177.             else  // C
178.             {
179.                 cur_pan_len = arrayA[topA - 1] * 2 + 1;
180.                 src_x = 11 - arrayA[topA - 1] + 66;
181.                 src_y = 14 - topC;
182.                 dst_x = src_x - 66;
183.                 dst_y = 15 - topA;
184.             }
185.         }
186.         else if (dst == 'B')
187.         {
188.             if (src == 'A')
189.             {
190.                 cur_pan_len = arrayB[topB - 1] * 2 + 1;  // topX 现有圆盘数量
191.                 src_x = 11 - arrayB[topB - 1];
192.                 src_y = 14 - topA;
193.                 dst_x = src_x + 33;
194.                 dst_y = 15 - topB;
195.             }
196.             else  // C
197.             {
```

```
198.                cur_pan_len = arrayB[topB - 1] * 2 + 1;
199.                src_x = 11 - arrayB[topB - 1] + 66;
200.                src_y = 14 - topC;
201.                dst_x = src_x - 33;
202.                dst_y = 15 - topB;
203.            }
204.        }
205.        else
206.        {
207.            if (src == 'A')
208.            {
209.                cur_pan_len = arrayC[topC - 1] * 2 + 1;  // topX 现有圆盘数量
210.                src_x = 11 - arrayC[topC - 1];
211.                src_y = 14 - topA;
212.                dst_x = src_x + 66;
213.                dst_y = 15 - topC;
214.            }
215.            else  // B
216.            {
217.                cur_pan_len = arrayC[topC - 1] * 2 + 1;
218.                src_x = 11 - arrayC[topC - 1] + 33;
219.                src_y = 14 - topB;
220.                dst_x = src_x + 33;
221.                dst_y = 15 - topC;
222.            }
223.        }
224.        GoToHigh(src_x, src_y, dst_x, dst_y, id, cur_pan_len);
225.        GoToInLine(src_x, src_y, dst_x, dst_y, id, cur_pan_len);
226.        GoToLow(src_x, src_y, dst_x, dst_y, id, cur_pan_len);
227.        cct_setcolor(); //恢复缺省颜色
228.    }
229.    void GoToInLine(int one_x, int one_y, int another_x, int another_y, int id, int
    len)  // 平移
230.    {
231.        const char ch = ' ';
232.        int bg_color = id;  // 保证移动时颜色不变
233.        const int fg_color = COLOR_BLACK;
234.        // 向左或向右移动
235.        if (one_x < another_x)  // 右移
236.        {
237.            const int MAX_X = another_x;
238.            for (int x = one_x; x <= MAX_X; x++)
239.            {
240.                cct_showch(x, 1, ch, bg_color, fg_color, len);
241.                if (move_speed != 0)
242.                    Delay(move_speed);
243.                else  // move_speed 选择为 0 时，每一步自动进行
244.                    Delay(4);
245.                cct_showch(x, 1, ch, COLOR_BLACK, COLOR_WHITE, len);  // 清除显示
246.                if (move_speed != 0)
247.                    Delay(move_speed);
248.                else  // move_speed 选择为 0 时，每一步自动进行
249.                    Delay(4);
250.            }
251.        }
252.        else  // 左移
253.        {
254.            const int MIN_X = another_x;
255.            for (int x = one_x; x >= MIN_X; x--)
256.            {
```

```
257.                 cct_showch(x, 1, ch, bg_color, fg_color, len);
258.                 if (move_speed != 0)
259.                     Delay(move_speed);
260.                 else  // move_speed 选择为 0 时，每一步自动进行
261.                     Delay(4);
262.                 cct_showch(x, 1, ch, COLOR_BLACK, COLOR_WHITE, len);  // 清除显示
263.                 if (move_speed != 0)
264.                     Delay(move_speed);
265.                 else  // move_speed 选择为 0 时，每一步自动进行
266.                     Delay(4);
267.             }
268.         }
269.     }
```

6.4 头文件

```
1.  void to_be_continued(const int X, const int Y);  // 一轮游戏结束后光标移动
2.  int Menu();  // 显示菜单，并返回用户正确选项
3.  void hanoi(int n, char src, char tmp, char dst, int choice);  // 递归函数
4.  void DoByChoice(int n, char src, char tmp, char dst, int choice);   // 根据用户选项来决定执行何种操作 选项：1、2、3、4、8
5.  void DoByChoice2(int n, char src, char tmp, char dst, int choice);  // 根据用户选项来决定执行何种操作 选项：6、7
6.  void DoByChoice3(int n, char src, char dst);                        // 根据用户选项来决定执行何种操作 选项：9
7.  void GuiLing();                                                     // 每次执行完一次用户选项后，静态全局变量置原值
8.  int IsMove(char from_ch, char to_ch);  // 选项 9 判断要不要移动 返回 0 表示会大盘压小盘，返回-1 表示起始柱为空，返回大于 0 表示要移动盘的编号
9.  void MovePanInArray(int n, char src, char dst);                       // 将盘子从 src 对应的数组移动到 dst 对应的数组
10. void PrintByChoice(int n, char src, char tmp, char dst, int choice);  // 紧接上一个函数，在屏幕上显示相应内容
11. void MovePanFromSrcToDst(int id, char src, char dst);
12. void GoToHigh(int one_x, int one_y, int another_x, int another_y, int id, int len);  // 上移
13. void GoToInLine(int one_x, int one_y, int another_x, int another_y, int id, int len);  // 平移
14. void GoToLow(int one_x, int one_y, int another_x, int another_y, int id, int len);  // 下移
15. void InputSrcDst(int* n_pre, char* src_ptr, char* tmp_ptr, char* dst_ptr);  // 输入盘子个数、起始柱和目标柱
16. void InputMoveSpeed();                                                      // 输入移动速度
17. void InitSrcArray(int n, char src);          // 初始化：往起始柱填盘子编号；越大的盘子，编号越大
18. void InitForOp4(int n, char src, char dst);  // 输出选项 4 的界面：第一行显示内容、纵向打印、横向打印
19. void InitForOp8(int n, char src, char dst);  // 输出选项 8 的界面：第一行显示内容、纵向打印、横向打印 + 圆柱 + 起始柱
20. void DrawTriYuanZhu();                // 画三根圆柱
21. void DrawSrcPanZi(int n, char src);  // 画初始化时起始柱上的盘子
22. void PrintInX();                              // 打印横向数组
23. void PrintInY(int offset_x, int offset_y);  // 打印纵向数组  不包括底座(A.B.C 和一排等号)
24. void Delay(int x);  // 延时设置
```