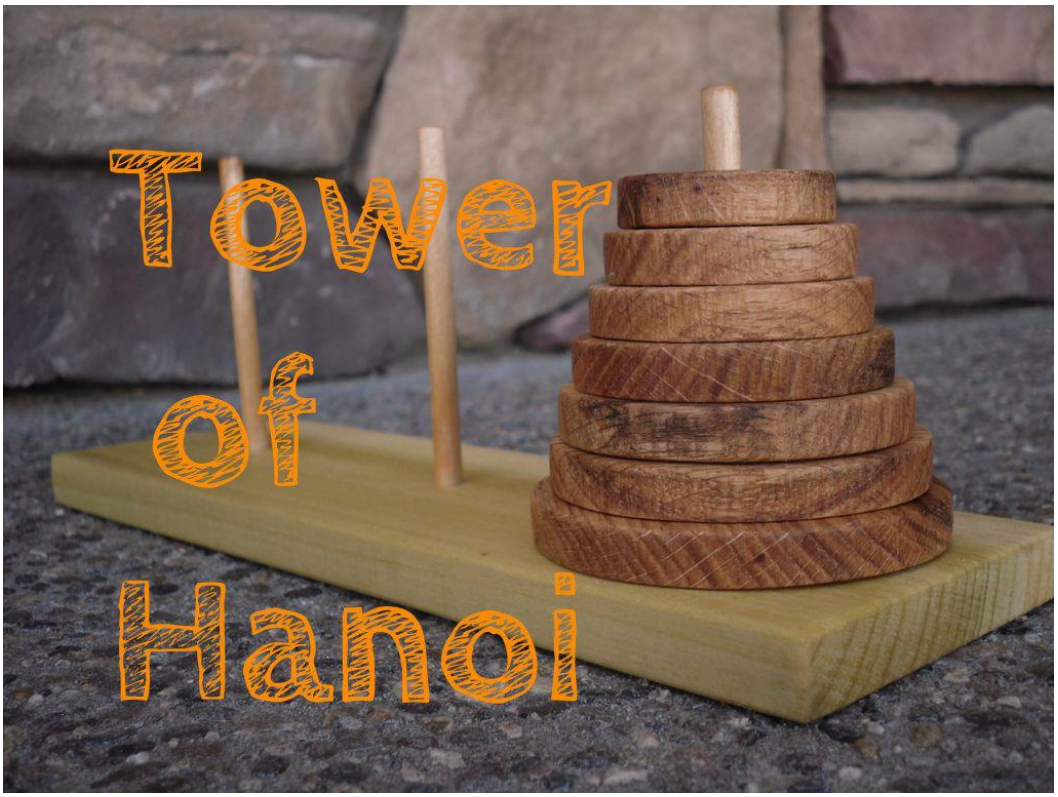
汉诺塔综合演示实验报告



2251079　隋建政
国豪06班
2022.11.30

1. 题目

通过菜单栏让用户进行选择，根据用户的选择进入到不同的模式。前四项为之前完成的汉诺塔显示模式，后续三项依次完成最终模式的准备工作，包括在屏幕上打印圆柱、打印相应的圆盘、完成圆盘的移动。第八项整合汉诺塔与准备程序，完成图像显示汉诺塔自动移动的模式。第九种为用户自行移动汉诺塔，当游戏成功后会退出该模式。任一模式完成后，会返回菜单栏，令用户再一次进行选择。

2. 整体设计思路

2.1 代码分布

调用函数hanoi_menu实现菜单栏在屏幕的输出，令用户进行选择，并将用户的选择返回给主函数的参数option，在主函数中根据option的不同取值进入到hanoi_multiple_solutions.cpp中的不同函数，以完成各个模式的实现。

2.2 函数的调用

在hanoi_multiple_solutions.cpp中，包含输入数据的shuru函数、实现汉诺塔算法的hanoi函数（唯一的递归函数）、打印汉诺塔步骤的print函数、横向输出函数及纵向输出函数、打印圆柱的column函数、打印初始盘子分布的plate函数、圆盘移动的move函数以及游戏模式的mode9函数。

选项1-4及6-9都需要调用shuru函数，其中1-4及和8都需要调用hanoi递归函数，再根据参数的不同调用不同的输出函数，以达到不同的效果。选项5单独调用column函数，选项7利用column函数并调用plate函数，选项7则是在6的基础上增添移动的效果，即再调用move函数。选项8在hanoi函数的基础上调用选项5-7的函数，选项9则是在mode9函数的基础上调用选项5-7的函数。

3. 主要功能的实现

3.1 菜单栏

在main函数中调用hanoi_menu函数，该函数的返回值为int类型，并将返回值赋值给参数option。

首先打印菜单栏，然后利用_getch()读入用户的选项，若用户输入的选项在0到9之间，则显示该选项，并继续之后的程序；若用户输入的选项超出预期范围，则不会显示输入结果并继续等待输入。

3.2 输入函数

在选项1、2、3、4、6、7、8、9中，均需调用shuru函数让用户进行初始数据的输入，依次输入圆盘个数（1-10）、起始柱（A-C）以及目标柱（A-C），在选项4和8中还需输入延迟时间参数

delay（0-5），该过程需要调用函数shuru_delay。

利用while循环和cin.good()进行判断可规避错误输出，当输出错误时，例如该输入数字时输入字母或输入的内容不在应有的范围内，便会清除缓冲区，并重新进入循环令用户重新进行输入；再如起始柱与目标柱相同时，会输出一行文字提示用户起始柱与目标柱相同，并令用户重新输入。当用户输入正确时，便会直接退出循环进行下一项输入或进入输出界面。而不论用户输入的是大写字母A-C还是小写字母a-c，程序均会按照大写字母A-C来处理。

该函数把指针作为参量，通过在函数中使用指针参数，以此实现通过函数影响主函数中实参的目的。

shuru函数的定义：

```
void shuru(int* n, char* src, char* tmp, char* dst,int option)
```

3.3 各个选项的实现

3.3.1 基本解

直接调用hanoi递归函数，并将option参数传给print函数使其完成输出。以此达到输出逐个步骤的目的。

3.3.2 基本解（步数记录）

在3.2.1的基础上，利用全局变量t记录步数，每输出一步便进行一次t++操作。以此达到输出步数及逐个步骤的目的。

3.3.3 内部数组显示（横向）

在3.2.2的基础上，根据等于3的option参数，输出每一步骤时同时调用shuchu3函数，即即时打印各个圆柱上的圆盘。

将各个圆柱上的圆盘以全局二维数组pla[3][10]记录，每利用hanoi函数计算一部移动，便将该步骤的起始柱与目标柱作为参数传递给shuchu3函数，在shuchu3函数中，使起始柱最顶上的盘子移动至目标柱最顶上盘子的上面一个，二维数组中的数据便代表各个盘子的编号。

然后依次将A、B、C三柱中记录的数组打印，若值不为0则输入该值并打印亿个空格，若该值为0则打印两个空格，这样既保证了圆盘移动后新的数组输出会将原先的输出完全覆盖，也保证了三个柱的输出长度保持不变。

3.3.4 内部数组显示（横向加纵向）

在3.2.3的基础上，根据等于4的参数option，调用shuchu4函数，在窗口中央位置打印内部数组的纵向显示。

具体实现方法为从下到上依次打印相应的pla[][0]到pla[][9]的值，若该值不为0，则直接打印该值；若该值为0，则打印的一个空格，每进行一次打印，便将纵坐标减一后继续打印。一整个圆柱打印完后，便使用cct_gotoxy()函数移动到下一个需要打印的圆柱位置。打印空格保证了圆盘移动后新的数组纵向输出会覆盖原先的输出。

需要注意的是，在该函数的输出中，不止要打印每个移动步骤后pla数组的横向与纵向显示，

还需要打印初始时pla数组的横向与纵向输出，在此我将其写为单独的beginning()函数进行输出。

其次，在该模式的实现中，需要设置延迟时间即delay全局变量，在shuru函数中调用输入该参量的shuru_delay()函数,通过利用系统函数Sleep()实现延时，参数设置为1000/delay，若delay为0，则按回车继续。

3.3.5 打印三根柱子

直接调用column()函数，该函数在窗口的指定位置打印三根柱子。

打印步骤为：在指定位置分别先画好三个基座，利用cct_showch()函数，将长度设置为23，将前景色和背景色均设置为14（亮黄色），打印完基座后期，便开始打印圆柱，每在一个指定位置打印完一个颜色设置为亮黄色且长度为1的小色块，便将纵坐标减一继续打印，打印完一根圆柱后，便转移至指定位置（x+=32），打印下一根圆柱，为了方便观察，利用Sleep()函数进行延时操作，将参数设置为50。

当三根圆柱全部打印完成后，要记得利用cct_setcolor()和cct_getcolor()函数将前景色和背景色设置为初始的黑底白字。

为了保证圆柱打印时的美观，在打印之前要利用函数cct_setcursor(CURSOR_INVISIBLE)隐藏光标，不然光标会跟随色块的打印一直闪烁，影响观感。

3.3.6 打印圆盘

先调用column()函数，再调用plate函数，使用的参数为层数n和起始柱的编号src，对于一个确定的src参数，可以找到该起始柱相对应的位置并从下到上依次打印由大到小的n个盘子。该过程利用for循环完成。

每一个圆盘的长度和颜色都仅由该圆盘本身的编号决定。

打印圆盘时也需要使用cct_setcursor(CURSOR_INVISIBLE)函数来隐藏光标。

3.3.7 第一次移动

先用column()函数和plate()函数画出圆柱和初始的圆盘，再用move()函数进行移动。move()函数可大体分为三个部分：圆盘上移、圆盘横向移、圆盘下移。

该函数所需的参数为圆盘编号、起始柱、目标柱、起始柱盘数、目标柱盘数以及延时时间，由于本模式不需要delay参数的输入因此设置为0。在该模式下，起始柱盘数即为用户设置的初始盘数，目标柱盘数就是0。由于本模式仅是实现第一次的移动，同时输入的参数只是最初的起始柱与最终的目标柱，为了区分3.3.7与3.3.8中的模式，还设置了进行判断的参数k，在本模式下可以观察到若盘数n为奇数，第一次移动的目标柱即为最终的目标柱；而若盘数n为偶数，那么第一次移动的目标柱并非最终的目标柱而是除初始柱和目标柱之外的中间柱，以此根据判断，若为模式7则k值为1，会进行目标柱的修订；否则k值设置为0，就不会进行这种修订。

移动操作的实现：

1） 上升时，在要移动圆盘的上一格位置打印一个相同的圆盘，由于圆盘的颜色和大小都由圆盘编号决定，这一操作很好实现，同时在原位置打印相同大小的黑色色块，并打印长度为1的亮黄色色块，以覆盖原先的圆盘使其成为空柱的模样。要注意的是，当圆盘上升至越

过圆柱顶端上一个的时候，就不应该再打印中间的亮黄色色块了，否则会出现圆柱延长的图样，该特例通过if判断圆盘纵坐标来规避。

2） 圆盘横向移动时，实现的逻辑与上升时类似，区别在于移动后仅需在圆盘身后补上一个长度为一的黑色小色块即可。该移动的问题在于根据目标柱和起始柱的不同，会有向左移和向右移两种运动轨迹，由于屏幕中的三个圆柱依次代表A、B、C三柱，那么根据起始柱参数src和目标柱参数dst的差就可以区分这两种情况。

3） 下降的实现逻辑与上升基本一致，但还需要注意的一个问题是当圆盘在最上面时其下降是打印的黑色块可能会遮挡原本输出的文字，该细节也可用if判断圆盘纵坐标是否在最上面来规避。

为了方便观察，该函数设置默认的延时时间Sleep(50)，即使之后delay参数输入为0，也会有默认延时时间的存在。在最后要使用cct_setcolor()函数和cct_getcolor()函数来使得窗口的打印恢复默认的黑底白字。

3.3.8 自动移动版本

该模式的实现需要综合3.3.4与3.3.7中的程序。

该模式实现的基本顺序为：

1） 调用3.3.4中的beginning()函数打印初始的汉诺塔纵向与横向显示，并调用之前的column()函数和plate()函数打印初始的汉诺塔图样，需注意在该模式中，需将纵向与横向显示整体向下移动，不然两个输出将会重合，这一点在之后的移动步骤中也是一样的，此即为初始页面。

2) 利用hanoi函数计算步骤之后，进入shuru3函数进行pla数组中数值的调换并进行横向打印，再根据option参数调用move()函数，即可实现初始柱最顶上一个圆盘向目标柱移动并落下的动画演示，之前的k值设置为0即可。最后再进行内部数组的纵向显示。

3.3.9 游戏模式

该模式的初始页面与3.3.8一致，不同的是后续移动不需要使用hanoi()函数计算，而由用户使用键盘输入操作，于是引入函数mode9()供用户输入。

正确的输入格式为分别输入起始柱与目标柱并回车，若用户的输入超过a-c的范围则会清除输入内容并等待用户继续输入。若用户输入Q则会退出游戏，该功能通过函数的返回值实现，若输入Q则会返回1使主函数中的循环调用终止，并输出“游戏中止”。

在该模式中还需对其余错误输入进行处理，由于在玩家输入后还需调用之前使用的内部数组横向与纵向显示及图形输出函数，将一系列错误处理放在shuchu3()函数中处理，分别用参数top1和参数top2代表起始柱顶上圆盘编号和目标柱顶上圆盘编号，若没有圆盘则为零。通过if判断，若top1的值大于top2的值，则是“大盘压小盘”；若是top1为零，则是“柱源为零”，发生任何一种错误均不会进行相应的移动操作，并继续等待玩家输入。

在主函数中利用for循环判断目标柱对应二维数组中的数值情况，当满足所有编号均移动到目标柱时，便会停止循环，打印“游戏结束!!!”并等待回车继续。

3.4 循环实现多次游戏

当一个模式结束后，会令用户按回车继续，此时会重新进入循环并重复打印最一开始的菜单栏，在输出菜单栏之前要将之前的delay参数（延时时间）、t参数（记录步数）以及一些判断循环结束的函数重新赋值为初始值，否则将会影响到下一次使用程序的功能。

4. 调试过程碰到的问题

4.1 move()函数的问题

在使用move()函数时，通过循环打印多种不同色块以实现在屏幕上圆盘移动的效果，但在打印色块时，往往会在不希望出现黑色块和圆柱的部位出现这些色块，解决办法为通过if语句判断并即时调整坐标。

1) 圆盘上升时，判断是否越过圆柱顶端上一格的位置，如超过则只打印黑色色块而不打印亮黄色的圆柱色块。

2) 圆盘左右平移时，坐标的改变量会有所不同，由于将初始的坐标设置为即将移动圆盘最左端的坐标，在向不同方向移动时便会出现不同的结果。若圆盘向右移动，每次在x-1的位置输出长度为1的黑色色块即可；而若圆盘向左移动，则需要在x+2n+1的位置输出长度为1的黑色色块（n为圆盘的编号，2n+1为该圆盘的长度），此处根据初始柱减目标柱的值通过if条件判断分为两种情况进行输出。

3) 圆盘下降时，在最顶上的时候如果与其他位置相同会在y=0的位置打印一个有一定长度的黑色色块，这可能会遮挡在第一行输出的文字，因此通过if判断纵坐标是否等于1，当纵坐标等于1时不打印该黑色色块。

4) 参数调用问题，由于在3.3.8和3.3.9中使用的move()函数与3.3.7中相同，在实现3.3.8和3.3.9的移动功能时需要更多的参数来进行，于是需要即时计算各个柱子上圆盘的数量，并根据这两个参数确定起始高度和终止高度，以实现圆盘移动的需要，并将3.3.7中的终止高度设为0，初始高度设置为盘数。

5) delay参数的设置，移动动画本身就应该带默认延时，延时时间的设置不能全用Sleep(50/delay)，不然当delay等于0时，不会演示圆盘移动的动画。此处应用if条件判断，若delay为0则使用默认的Sleep(50)。

4.2 输入过程中出现的问题

要进行输入时，在主函数中调用shuru()函数进行输入，由于输入的数据在主函数接下来的部

分还会进行使用，因此需要使用指针变量作为形参，以达到改变实参的目的。

当用户输入每一项后，都必须对其进行单独判断，若输入错误则需令用户再次输入。在此处需要注意的问题是若用户输入多余指定数量的数据，将有可能产生错误，如用户在输入起始柱时输入ac，那么目标柱dst会自动读入c，而不需要用户再进行输入，那么此时就需要使用cin.ignore()清除缓冲区中溢出的部分。而且还应注意的是，不论用户输入正确还是错误，都需要清除缓冲区，否则一样会产生错误。

4.3 3.3.9中出现的问题

在3.3.9中mode9会调用之前使用的shuchu3()函数，在判断用户错误步骤时会出现问题，如尽管报错，仍按用户指示进行移动；或者尽管报错且没有进行移动，但下一次在进行移动时却是在错误的基础上进行的。此时需要改变逻辑判断与执行移动部分之间的位置。若判断为错误步骤，则不会进行pla二维数组间的调换，且需要注意的是，在调用move()函数是，传入的num_src和num_dst参量应是移动前的各圆柱上的圆盘数量而非移动后的数量，因此应该先统计数量，再执行数据的调换。

5. 心得体会

5.1 经验教训

在调试中发现错误并打算修改时，可以先从代码中寻找细节上的错误，例如switch-case语句中有没有加break，或者某些参数有没有打错，再或者在复制粘贴改变代码格局时，有没有多出或漏掉一行。时常怀疑代码基本逻辑的正确性，只会给自己徒增焦虑。

在给函数命名时，应该将其命名的更加易读，看到函数名称就能回想起大致内容，这会给各个步骤间函数的调用带来便利，尤其是在完成这种调用函数较多的大作业中，清晰的命名可以防止遗忘节约时间。

5.2 程序拆分

对于一些较为复杂的程序，我认为还是拆分为一些小作业来进行比较好，分解为小作业，可以对各个功能的实现有更具体的认识，对各种细节问题的把握会更加具象。若一次性完成，则会感觉因为各种问题太过繁杂而无从下手。

在各个功能的实现上，可能会调用相同的函数，拆分为小作业来做，只需在应有的地方调用之前写过的函数即可。但仍须注意的是，由于功能不完全相同，传递给函数的参数不会完全相同，如果在之后的调用中还需要更多的参数，也不需要重写函数，直接增添函数定义中参数的数量的即可，还要记得把原先参数更少的函数中本不需要用到的参数设置为相应的值。

5.3 前后程序的联系

在完成后面小题的过程中会利用到前面小题的结果，前后小题之间大部分是有联系的，在实 现类似功能时，只需要传递相应的参数并按着一定顺序调用可以利用的之前已完成的函数即可。

为了更好的利用已完成的代码，应寻找功能实现上的共性，通过使用不同的参数修订在各个模式之间的差异，也应注意在最初模式函数参数的使用，不然也会产生错误。

5.3 函数的使用

在程序中调用函数，可以有效降低工作量和代码的数量，将经常重复的内容写作函数，可以减少代码的重复，增强代码的可读性。把实现不同功能的代码写为不同的函数，可以使代码的结构更加清晰，也方便理解与修改。且函数之间可能会互相调用，函数的使用是这一过程得以轻松的实现。

如何利用好函数这一有效的工具，应该先把主函数中的大体脉络理清，再插入不同的函数以实现不同的功能。函数命名要清晰，减少不必要的记忆量，也方便后续过程的调用。再就是函数参数的传递，不同功能可能需要不同的参数，即时不在第一时间完全想清楚所有的功能也没关系，在后续代码中加入函数即可，注意在增添参数后不要忘记把之前功能的函数调用增添相应的参数。

6. 附件：源程序

```
#include <iostream>
#include <conio.h>
#include "hanoi.h"
using namespace std;
int t=1, col[3], pla[3][10],delay=0;
int main()
{
    int option,n=0,i,ret,k=0,x=0,m;
    char src=0, tmp=0, dst=0;
    while (1) {
        t = 1;
        delay = 0;
        k = 0;
        cct_cls();
        option = hanoi_menu();
        if (option == 0)
            break;
        if (option == 1 || option == 2 || option
== 3
        || option == 4 || option == 6 || option
== 7
        || option == 8||option==9) {
            shuru(&n, &src, &tmp, &dst,
option);
            if (option != 6 && option != 7) {
                if (option == 4 || option == 8
 || option == 9)
                    print4(src, tmp, dst, n,
option);
                else
                    hanoi(n, src, tmp, dst,
option);
            }
        }
        if (option == 5 || option == 6
          || option == 7 ) {
            cct_cls();
            column();
        }
        if (option == 6 || option == 7)
            plate(n, src);

        if (option == 7)
            move(1, src, dst, n, 0, 1,0);
        if (option == 9) {
            while (!k) {
                m=mode9(n, option);
                if (m)
                    break;
                for (i = 0; i < 10; i++) {
                    k = 1;
                    switch (dst) {
                        case 'A':
                            x = 0;
                            break;
                        case 'B':
                            x = 1;
                            break;
                        case 'C':
                            x = 2;
                            break;
                    }
                    if (pla[col[x]][i] != n - i
                            && n - i > 0) {
                        k = 0;
                        break;
                    }
                }
            }
            for (i = 0; i < 7; i++)
                cout << endl;
            if (m)
                cout << "游戏中止!!!" << endl;
            else
                cout << "游戏结束!!!" << endl;
            cout << endl;
        }
        if(option!=9)
            for (i = 0; i < 10; i++)
                cout << endl;
        cct_setcursor(CURSOR_VISIBLE_NORMAL);
        cout << "按回车继续";
        while (1) {
            ret = _getch();
            if (ret == 13)
                break;
        }
    }
    return 0;
}

extern int t, col[3], pla[3][10],delay;
void column()
{
    int x, y;
    int bg_color, fg_color;
    cct_setcursor(CURSOR_INVISIBLE);
    cct_showch(1, 15, 'x', 14, 14, 23);
    cct_showch(33, 15, 'x', 14, 14, 23);
    cct_showch(65, 15, 'x', 14, 14, 23);
    for (y = 14; y >= 3; y--) {
        for (x = 12; x <= 76; x += 32) {
            Sleep(50);
            cct_showch(x, y, 'x', 14, 14, 1);
```

```
        }
    }
    cct_setcolor(0, 7);
    cct_getcolor(bg_color, fg_color);
}
void plate(int n, char src)
{
    int x = 0, y = 14, i;
    int bg_color, fg_color;
    cct_setcursor(CURSOR_INVISIBLE);
    x = 12 + 32 * (src - 65);
    for (i = n; i > 0; i--) {
        Sleep(50);
        cct_showch(x - i, y, 'x', i, i, 2 * i + 1);
        y--;
    }
    cct_setcolor(0, 7);
    cct_getcolor(bg_color, fg_color);
}
void move(int n, char src, char dst, int
num_src,int num_dst,int k,int delay)
{
    int x, y;
    int bg_color, fg_color;
    x = 12 + 32 * (src - 65) -n;
    if (delay)
        Sleep(1000 / delay);
    else
        Sleep(300);
    cct_setcursor(CURSOR_INVISIBLE);
    for (y = 14 - num_src; y >= 1; y--) {
        if (delay)
            Sleep(50 / delay);
        else
            Sleep(50);
        cct_showch(x, y, 'x', n, n, 2 * n + 1);
        cct_showch(x, y + 1, 'x', 0, 0, 2 * n + 1);
        cct_showch(x + n, y + 1, 'x', 14, 14, 1);
        if (y == 1)
            cct_showch(x + n, y + 1, 'x', 0, 0, 1);
    }
    y++;
    if (k)
        if (num_src % 2 == 0)
            dst = 198 - dst - src;
    if (dst > src) {
        for (x++; x != 12 + 32 * (dst - 65)-n+1;
x++) {
            if (delay)
                Sleep(50 / delay);
            else
                Sleep(50);
            cct_showch(x, y, 'x', n, n, 2 * n + 1);
            cct_showch(x - 1, y, 'x', 0, 0, 1);
        }
        x--;
        y++;
    }
    else {
        for (x--; x != 12 + 32 * (dst - 65) - n-1;
x--) {
            if (delay)
                Sleep(50 / delay);
            else
                Sleep(50);
            cct_showch(x, y, 'x', n, n, 2 * n + 1);
            cct_showch(x + 2 * n + 1, y, 'x', 0,
0, 1);
        }
        x++;
    }
    for (; y <= 14-num_dst; y++) {
        if (delay)
            Sleep(50 / delay);
        else
            Sleep(50);
        cct_showch(x, y, 'x', n, n, 2 * n + 1);
        if (y != 1) {
            cct_showch(x, y - 1, 'x', 0, 0, 2 * n
+ 1);
            cct_showch(x + n, y - 1, 'x', 14, 14,
1);
        }
        if (y <= 3&&y!=1)
            cct_showch(x + n, y - 1, 'x', 0, 0, 1);
    }
    cct_setcolor(0, 7);
    cct_getcolor(bg_color, fg_color);
}
void shuchu3(int n, char src, char dst, int option)
{
    int i, x = 0,top1=0,top2=0;
    int num[3];
    for (i = 0; i < 3; i++)
        num[i] = 0;
    for (i = 0; i < 10; i++)
        if (pla[col[src-'A']][i] != 0)
            num[col[src-'A']]++;
    for (i = 0; i < 10; i++)
        if (pla[col[dst-'A']][i] != 0)
            num[col[dst-'A']]++;
    for (i = 0; i < 10; i++)
        if (pla[col[198-dst-src-'A']][i] != 0)
            num[col[198-dst-src-'A']]++;
    for (i = 0; i < 10; i++) {
        if (pla[col[src - 'A']][i] == 0)
            break;
        else
            top1 = pla[col[src - 'A']][i];
    }
    for (i = 0; i < 10; i++) {
        if (pla[col[dst - 'A']][i] == 0)
            break;
        else
```

```
            top2 = pla[col[dst - 'A']][i];
    }
    if (option == 9) {
        cct_gotoxy(0, 37);
        cout << "第" << setw(4) << t << " 步(" <<
top1 << "#): " << src << "-->" << dst;
    }
    if (option != 9 || (option == 9 && (num[col[src
- 'A']] != 0 && (top1 < top2 || top2 == 0)))) {
        for (i = 0; i < 10; i++) {
            if (pla[col[src - 65]][i] == 0 && i !=
0) {
                x = pla[col[src - 65]][i - 1];
                pla[col[src - 65]][i - 1] = 0;
                break;
            }
            else if (i == 9) {
                x = pla[col[src - 65]][i];
                pla[col[src - 65]][i] = 0;
                break;
            }
        }
        for (i = 0; i < 10; i++)
            if (pla[col[dst - 65]][i] == 0) {
                pla[col[dst - 65]][i] = x;
                break;
            }
    }
    cout << " A:";
    for (i = 0; i < 10; i++) {
        if (pla[col[0]][i] == 0)
            cout << "  ";
        else
            cout << " " << pla[col[0]][i];
    }
    cout << " B:";
    for (i = 0; i < 10; i++)
        if (pla[col[1]][i] == 0)
            cout << "  ";
        else {
            cout << " " << pla[col[1]][i];
        }
        cout << " C:";
    for (i = 0; i < 10; i++)
        if (pla[col[2]][i] == 0)
            cout << "  ";
        else{
            cout << " " << pla[col[2]][i];
        }
    if (option == 8||option==9) {
        if (option == 9 && num[col[src - 'A']] ==
0) {
            cout << endl;
            cout << endl;
            cout << endl;
            cout << "柱源为零";
            Sleep(500);
        }
        else if (option==9&&top1 > top2&&top2!=0)
       {
            cout << endl;
            cout << endl;
            cout << endl;
            cout << "大盘压小盘，非法移动";
            Sleep(500);
        }
        else {
            move(top1, src, dst, num[col[src -
'A']], num[col[dst - 'A']], 0, delay);
            if (option == 9)
                t++;
        }
    }
}
void shuchu4(int n,char src,char dst,int option)
{
    int ret,x=11,y=11,i;
    if (option == 8||option==9)
        y = 31;
    if (option != 9) {
        cct_gotoxy(0, 17);
        if (option == 8)
            cct_gotoxy(0, 37);
        cout << "第" << setw(4) << t << " 步(" <<
n << "#): " << src << "-->" << dst;
    }
    shuchu3(n,src, dst,option);
    cct_gotoxy(9, 12);
    if (option == 8||option==9)
        cct_gotoxy(9, 32);
    cout << "=======================";
    cct_gotoxy(x, y);
    for (i = 0; i < 10; i++) {
        if (pla[col[0]][i] == 0)
            cout << ' ';
        else {
            cout << pla[col[0]][i];
            y = y - 1;
            cct_gotoxy(x, y);
        }
    }
    x = 21;
    y = 11;
    if (option == 8||option==9)
        y=31;
    cct_gotoxy(x, y);
    for (i = 0; i < 10; i++) {
        if (pla[col[1]][i] == 0)
            cout << ' ';
        else {
            cout << pla[col[1]][i];
            y = y - 1;
            cct_gotoxy(x, y);
        }
```

```
    }
    x = 31;
    y = 11;
    if (option == 8||option==9)
        y = 31;
    cct_gotoxy(x, y);
    for (i = 0; i < 10; i++) {
        if (pla[col[2]][i] == 0)
            cout << ' ';
        else {
            cout << pla[col[2]][i];
            y = y - 1;
            cct_gotoxy(x, y);
        }
    }
    cct_gotoxy(11, 13);
    if (option == 8||option==9)
        cct_gotoxy(11, 33);
    cout << "A         B         C" << endl;
    if (option != 9) {
        if (!delay)
            while (1) {
                ret = _getch();
                if (ret == 13)
                    break;
            }
        else
            Sleep(1000 / delay);
    }
}
void print(int n, char src,char dst,char option)
{
    if (option == 1) {
        cout << n << "#" << " " << src << "---->"
<< dst << endl;
        t++;
    }
    if (option == 2||option==3) {
        cout << setiosflags(ios::right);
        cout << "第" << setw(4) << t << " 步( " <<
n << "#: " << src << "-->" << dst << ")";
        if (option == 3)
            shuchu3(n,src,dst,option);
        cout << endl;
        t++;
    }
    if (option == 4||option==8) {
        shuchu4(n,src,dst,option);
        t++;
    }
}

void hanoi(int n, char src, char tmp, char dst,int
option)
{
    if (n == 1)
        print(n, src, dst, option);
    else {
        hanoi(n - 1, src, dst, tmp,option);
        print(n, src, dst, option);
        hanoi(n - 1, tmp, src, dst,option);
 }
    cct_setcursor(CURSOR_VISIBLE_NORMAL);
}
void beginning()
{
    int i;
    cout << "初始: A:";
    for (i = 0; i < 10; i++) {
        if (pla[col[0]][i] == 0)
            cout << "  ";
        else
            cout << " " << pla[col[0]][i];
    }
    cout << " B:";
    for (i = 0; i < 10; i++) {
        if (pla[col[1]][i] == 0)
            cout << "  ";
        else
            cout << " " << pla[col[1]][i];
    }
    cout << " C:";
    for (i = 0; i < 10; i++) {
        if (pla[col[2]][i] == 0)
            cout << "  ";
        else
            cout << " " << pla[col[2]][i];
    }
}

void print4(char src,char tmp,char dst,int n,int
option)
{
    int ret,i,x=11,y=11;
    if (option == 8 || option == 9)
        y += 20;
    cct_cls();
    if(option!=9)
        cout << "从" << src << "移动到" << dst <<
", 共" << n << "层, 延时设置为" << delay << endl;
    else
        cout << "从" << src << "移动到" << dst <<
", 共" << n << "层" << endl;
    cct_gotoxy(0, 17);
    if (option == 8 || option == 9)
        cct_gotoxy(0, 37);
    beginning();
    cct_gotoxy(9, 12);
    if (option == 8 || option == 9)
        cct_gotoxy(9, 32);
    cout << "=======================";
    cct_gotoxy(x, y);
    for (i = 0; i < 10; i++) {
        if (pla[col[0]][i] == 0)
```

```
            break;
        else {
            cout << pla[col[0]][i];
            y = y - 1;
            cct_gotoxy(x, y);
        }
    }
    x = 21;
    cct_gotoxy(x, y);
    for (i = 0; i < 10; i++) {
        if (pla[col[1]][i] == 0)
            break;
        else {
            cout << pla[col[1]][i];
            y = y - 1;
            cct_gotoxy(x, y);
        }
    }
    x = 31;
    cct_gotoxy(x, y);
    for (i = 0; i < 10; i++) {
        if (pla[col[2]][i] == 0)
            break;
        else {
            cout << pla[col[2]][i];
            y = y - 1;
            cct_gotoxy(x, y);
        }
    }
    cct_gotoxy(11, 13);
    if (option == 8 || option == 9)
        cct_gotoxy(11, 33);
    cout << "A         B         C" << endl;
    if (option == 8||option==9) {
        column();
        plate(n, src);
    }
    if (option != 9) {
        if (!delay)
            while (1) {
                ret = _getch();
                if (ret == 13)
                    break;
            }
        else
            Sleep(1000 / delay);
    }
        hanoi(n, src, tmp, dst, option);
}
    cout << "[请选择:]";
    while(1) {
         option = _getch();
         if (option >= '0' && option <= '9') {
              cout << char(option) << endl;
              break;
         }
    }
```

```
int mode9(int n,int option)
{
    char src_9, dst_9;
    cct_setcursor(CURSOR_VISIBLE_NORMAL);
    while (1) {
        cct_gotoxy(0, 39);
        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)：
";
        cout << endl;
        cout                    <<                    "
";
        cct_gotoxy(60, 39);
        cin >> src_9;
        if (src_9 == 'Q'||src_9=='q')
            return 1;
        else {
            cin >> dst_9;
            if (src_9 >= 'a' && src_9 <= 'c')
                src_9 -= 32;
            if (dst_9 >= 'a' && dst_9 <= 'c')
                dst_9 -= 32;
            if (src_9 >= 'A' && src_9 <= 'C' &&
dst_9 >= 'A' && dst_9 <= 'C' && dst_9 != src_9)
                break;
        }
    }
    cin.ignore();
    shuchu4(n, src_9, dst_9, option);
    return 0;
}
int hanoi_menu()
{
     int option;
     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;
     cout << endl;
     return option-48;
}
```