汉诺塔综合演示
HANOI TOWER

姓　　名：苗君文

学　　号：2253893

班　　级：信 06

完成日期：2023 年 5 月 21 日

二〇二三年五月

1. 题目：汉诺塔综合演示

汉诺塔（Hanoi Tower），又称河内塔，是一个源于印度古老传说的益智玩具。大梵天创造世界的时候做了三根金刚石柱子，在一根柱子上从下往上按照大小顺序摞着64片黄金圆盘。大梵天命令婆罗门把圆盘从下面开始按大小顺序重新摆放在另一根柱子上。并且规定，在小圆盘上不能放大圆盘，在三根柱子之间一次只能移动一个圆盘。

根据这个传说，便产生了经典的汉诺塔问题：有三根相邻的柱子，标号为A,B,C，选择一根起始柱（例如：A）和一根目标柱（例如：B），A柱子上从下到上按金字塔状叠放着n个不同大小的圆盘，要把所有盘子一个一个移动到目标柱B上，并且每次移动同一根柱子上都不能出现大盘子在小盘子上方，移动过程中允许在三根柱子之间任意进行。请问至少需要移动多少次才能使所有盘子按要求移到目标柱上？

正因汉诺塔问题蕴含着递归思维，汉诺塔问题也是程序设计中的经典递归问题。本题要求设计6种展现汉诺塔的方式（要求每种方式均考虑输入错误，其中5-7为8与9的分解步骤）：

1. 汉诺塔的基本解（仅展示移动步骤）
2. 汉诺塔的基本解（展示移动步骤并记录步数）
3. 显示汉诺塔内部数组（仅展示横向）
4. 显示汉诺塔的内部数组（展示横向与纵向）
5. 图形解-预备-画三个圆柱
6. 图形解-预备-在起始柱上画n个盘子
7. 图形解-预备-展现第一次移动
8. 图形解-展现汉诺塔以图形样式自动移动
9. 图形解-游戏版汉诺塔

此外，本题要求将6种汉诺塔的展现形式集成在一个程序中，并通过菜单方式来选择。

2. 整体设计思路

先展示菜单，再选择其中某一个选项，进入并运行，若为选项1-9，运行完则回到菜单，由此可见整个程序结构为循环结构。并将这个结构放在hanoi_main.cpp文件中。

展示菜单的方式通过调用menu函数，为使整个程序分布清晰明辨，将menu函数单独放在

hanoi_menu.cpp文件中，并将该函数定义为char型，在menu函数中显示菜单各项，并输入菜单选项，输入时作为字符型输入，从而返回该字符值。

进入选项0时，调用solution0函数，调用后通过break跳出循环，结束程序。而进入选项1-9时，分别调用solution1-solution9的函数。所有菜单选项函数，即solution0-9均放在hanoi_multiple_solution.cpp文件。

为调用以上函数，通过在hanoi_main.cpp文件中先申明各函数来完成。

3. 主要功能的实现

此处将介绍solution0-solution9各函数的实现。

3.1.solution0（退出）的实现

将光标移至（0,38）后跳出循环。

3.2.solution1（基本解）的实现

通过hanoi_input函数输入多个参数，包括起始柱、目标柱及层数。而后进入hanoi递归函数，每次递归时进入hanoi_print_step函数，为根据需要调用所需内容，需要增加传参量“solution”，此处则传“1”。

hanoi_input函数：用来输入各种参数（包括汉诺塔的层数、起始柱、目标柱、及后续会用到的移动速度）；输入参数有int* p_n, char* p_src, char* p_tmp, char* p_dst,int solution，通过指针可以避免返回值只能回传一个参数。同时，该函数还可以通过src及dst算出tmp，根据solution的值来选择是否输入speed（后续选项将会使用到）。

每次输入都要考虑输入错误，通过循环，如果输入错误（包括输入类型错误及范围错误）则要求重新输入，解决输入错误的关键步骤为cin.clear();while ((getchar()) != '\n');然后通过continue继续循环；若输入正确则需清空缓冲区，也需通过cin.clear();while ((getchar()) != '\n');来解决。在正确输入src后，还需将小写字母的src转换为大写字母，方便后续的计算及代码书写。同时，解决dst的输入错误时，应特别注意目标柱不能与起始柱相同。正确输入dst后还应将小写字母转换为大写字母，方便后续计算。Solution值为4或8时需要输入speed。

hanoi_print_step函数：该函数是整个程序里输出的关键，用来打印所有步骤。输入参数有int n, char src, char tmp, char dst, int solution。将在每个不同的solution中分别阐述其不同的功用。

此处则是hanoi_print_step(n, src, tmp, dst, 1)，输出正在移动的盘号及其移动的路径，包含起始柱及目标柱。

hanoi函数：该函数为贯穿整个程序的核心函数，输入参数有int n, char src, char tmp, char dst, int solution，此处solution在该函数中作用虽未体现，但可以传递进hanoi_print_step函数，方便后续的调用。本函数中根据n的值分为n=1与n>1两种进行递归。

3.3.solution2（基本解-步数记录）的实现

与solution1函数相比，solution2增加了步数记录，即累计每一次移动的次数。则可以通过定义一个全局变量num来实现，初始时定义num值为0。并将num的增加放在hanoi_print_step函数中。在输出“第num步”之前进行num++以统计步数。

solution2的实现为通过hanoi_input函数输入多个参数，而后进入hanoi递归函数，每次递归时进入hanoi_print_step函数，输出选项2所对应的内容。

3.4.solution3（内部数组显示-横向）的实现

总体上，实现思路与前两个选项相同。通过hanoi_input函数输入多个参数，而后进入hanoi递归函数，每次递归时进入hanoi_print_step函数所对应solution3的内容。值得注意的是，每次进入solution4函数后需要通过reset_array函数重置数组，将数组中的所有元素都变为0，防止前一次进行汉诺塔的数组还留存在数组中，同时，也应重置三个栈顶指针，也将其置为零，否则会导致程序错误，无法实现。再输入各类参数后，需要通过initialize_top函数来初始化三个top值。此处依旧需要记录步数，与前者相同，使用全局变量num++来实现。但横向数组的输出由于后续都将用到，另外定义一个横向输出数组的函数output_heng来实现。

reset_array函数：用来将所有数组元素变为0，将三个top值（top1、top2、top3）置为0。由于三个数组均为全局数组，三个栈顶指针top均为全局简单变量，该函数无需增加任何输入参数。

initialize_top函数：用来初始化三个top值，及数组中的元素。输入参数为char *p_src, int *p_n。

output_heng函数：用来调整三个数组中各个盘子的位置，并横向输出各盘的变化。输入参数为int n, char src, char tmp, char dst, int solution。使用三个全局一维数组来记录A,B,C三个圆柱中的圆盘数及编号，再用三个全局简单变量表示A,B,C三个栈的栈顶指针。用一维数组来模拟一个栈，完成栈的基本操作，基本方式为：假设数组大小为10，把[0]作为栈底，用一个int top作为栈顶指针，指向元素即将插入的位置。通过栈的方式，来将起始柱数组中的盘子移动到目标柱数组中。而后输出横向的步数记录，通过调用print_abc_number函数来完成。

print_abc_number函数：用来输出当前数组中的盘号，并且按照所在位置横向输出。当数组元素值为0时，输出两个空格，不为0时，则右对齐输出宽度为2的圆盘编号。

3.5.solution4（内部数组显示-纵向+横向）的实现

通过hanoi_input函数输入多个参数，此处需要输入岩石设置的speed，而后进入hanoi递归函数，每次递归时进入hanoi_print_step函数所对应solution4的内容。值得注意的是，每次进入solution4函数后需要重置数组，将数组中的所有元素都变为0，防止前一次进行汉诺塔的数组还留存在数组中，同时，也应重置三个栈顶指针，也将其置为零，否则会导致程序错误，无法实现。由于solution4中需要展示纵向的数组变化，会有动画效果，因此在重置数组及栈顶指针、输入各类参数后需要隐藏光标让动画效果更为明显且美观。

为展示动画效果，输入完成后需要清屏，并将光标移动到（0,0）位置，输出“从src移动到dst，共k层，延时设置为speed”这样一句话，并且在移动过程中，这句话始终保持位置不变，并持续显示。而后，先输出初始状态，通过output_zong函数在指定位置打印纵向的数组盘数变化。然后不光标移至（0,19）位置打印横向的数组，此时，打印初始状态完成。

而后进入hanoi递归函数，打印移动的过程。进入hanoi函数后，则每一步需要进入hanoi_print_step函数打印步骤。对应solution4的内容为：先num++来记录步数，再打印横向和纵向的数组。由于此处有延时设置，需要分类讨论，当延时设置为0时，需要通过回车来进入到下一步。具体则是通过(_getch() == 13)来实现，设置一个循环，若键入的不是回车，则还循环进入，直至为回车键。当延时设置为1-5时，则应通过Sleep(500/speed)来延迟输出，从而实现动画效果。

Output_zong函数：输出纵向的数组。输入参数为int n,char src,char tmp,char dst,int solution,int base_x,int base_y，其中base_x,base_y为“A”字符的位置，在solution4中该坐标值为（10,15）。通过x改变，来移动到不同的柱子上输出，通过y的改变，来移动到不同的位置输出盘号。

3.6.solution5（图形解-预备-画三个圆柱）的实现

为实现8-9两个选项，5-7提供了铺垫。选项5要求画三个柱子，根据所给的tool可以解决。

具体实现为：首先清屏，隐藏光标，进入draw_column函数。此处是由于5-9均需要画柱子，因此共用一个函数解决。画好柱子后，需要恢复默认颜色，使用cct_setcolor()函数来实现，最后通过to_be_continued函数来结束选项5的实现。

Draw_column函数：用来画三个柱子。无需输入参数。定义x，y两个横纵坐标值，来锁定ABC三个柱子所在的位置。先画三个底座，使用cct_showch(x, y, ' ', COLOR_HYELLOW, COLOR_HYELLOW, 1);再向上画三根柱子，运用循环结构，从左至右，y的改变为外循环，x的改变为内循环。

To_be_continued函数：此处是仿照test-cct.cpp文件中的同名函数来写的，用来结束当前移动，并显示“按回车键继续”。

3.7.solution6（图形解-预备-在起始柱上画n个盘子）的实现

此函数的实现需要先置零数组及栈顶指针，再输入各类参数，而后初始化各个数组及栈顶指针top值。

输入完成后，清屏，先输出“从src移动到dst，共k层”，然后进入draw_column函数先画三个柱子，然后再进入draw_plates函数画起始柱上的n个盘子，最后通过to_be_continued函数结束该选项的进程。

draw_plates函数：用来画起始柱上的n个盘子。通过数组来画，因此无需输入参数。定义x，y两个横纵坐标值，来锁定ABC三个柱子所在的位置。同样地，隐藏光标，以更好的显示。根据数组元素的值，来画盘子。若有值，则使用cct_showch(x - s1[i] - 1 + j, y, ' ', s1[i], s1[i], 1);通过盘子的编号来定义盘子的颜色，由上到下画，若元素值为0，则画柱子，左右两边为黑色，中间为黑色。为实现动画效果，在画有颜色的盘子时，需要Sleep(20)来体现差异。

3.8. solution7（图形解-预备-第一次移动）的实现

调用draw_column，draw_plates，并通过层数来判断第一次移到目标柱还是中间柱。

3.9. solution8（图形解-自动移动版本）的实现

该选项是前几选项的组合。

3.10. solution9（图形解-游戏版）的实现

该选项无需递归函数，关键根据输入来完成函数。

4. 调试过程碰到的问题

4.1. 游戏版选项的输入问题

通过不同输入函数，最终通过定义一个数组letter[127]，并通过_getch函数来输入，经过判断完成。

4.2. 在选项8-自动移动版本中输出的问题

4.2.1. 先显示移动盘子再显示改变横纵输出的问题

经过多次调试，及输出中间变量，我找到了解决方法：通过先返回去一步数组，然后输出盘子的移动，移动完再前进一步数组，恢复到调用draw_moving_plates前的数组。我通过两个

函数back_array和ahead_array来完成。

4.2.2. 显示奇怪色块的问题

通过研究test-cct.cpp文件，得知是没有恢复初始颜色设置的问题，再每次改变颜色的函数后增加了setcolor函数。

5. 心得体会

5.1.经验教训

每次写小作业的时候都要认真写，及时订正。改正错误的程序。比如此次作业中选项4，我之前是通过清除整个屏幕来打印每一步的，而这次改了很久才完成选项4达到只是改变纵向数组而不清屏。由此可见，每一次好好听习题课并优化自己的代码是一件很重要的事情。

5.2.若干小题还是大题？

在做一些较为复杂的程序时，分为若干小题更好。如果让我直接做选项9就会完全没有思路，但是5-7给了我循序渐进去思考的过程，当我完成5-7选项的时候基本上就已经水到渠成了，选项8-9剩下来的部分只需要思考一会儿就能想通，由此可见，后续的编写程序过程中，学会自己拆解程序也是一件重要的事情。

5.3.关联性及有效利用

在写汉诺塔大作业时，文档中给了很多提示，让我能够有效的利用之前几次汉诺塔作业的源程序，但也显示出了我之前写程序的弊端，没有拆分每一个功能，而是将很多东西混在一起写，也导致我思路很乱。所以说，写程序思路最重要。每次写代码时，应该遇到一个功能就把它放到一个函数中。

5.4.如何更好利用函数？

针对不同的功能，编写不同的函数，从而调用。本作业中，从整体思路到每一个选项的具体执行，都是在进行不同函数的组装。在本次作业中，要求hanoi递归函数不超过15行也是更好的教会我们去调用函数，我使用了hanoi_print_step函数，以不同的solution来调用不同选项对应的输出内容。

6. 附件：源程序

6.1.hanoi.h

```
#pragma once
#include <iostream>
#include <cstdio>
#include <conio.h>
#include <iomanip>
```

```
#include<Windows.h>
#include"cmd_console_tools.h"
```

6.2.hanoi_main.cpp

```
#include"hanoi.h"
using namespace std;
char menu();
void solution0();
//申明void solution1-9()与0类似，略
int main()
{    cct_setconsoleborder(120, 40, 120, 9000);
    char x;
    while (1) {
        x = menu();
        if (x == '0') {
            solution0();
            break;
        }
        if (x == '1')
            solution1();
        //x为2-9的情况与1类似，略
    return 0;
}
```

6.3.hanoi_menu.cpp

```
#include"hanoi.h"
using namespace std;
char menu()
{    char x;
    cct_cls();
    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 << "[请选择:] ";
    cct_setcursor(CURSOR_VISIBLE_NORMAL);
    while (1) {
        x = _getch();
        if (x >= '0' && x <= '9') {
            cout << x << endl << endl << endl;
            break;
        }
        else
            cct_gotoxy(10, 12);
    }
    return x;
}
```

6.4.hanoi_multiple_solutions.cpp

```
#include"hanoi.h"
#include"hanoi.h"
using namespace std;
int num = 0, speed;
int top1 = 0, top2 = 0, top3 = 0;
int s1[10] = { 0 }, s2[10] = { 0 }, s3[10] =
{ 0 };
void hanoi_input(int* p_n, char* p_src, char*
p_tmp, char* p_dst,int solution)
{    cct_setcursor(CURSOR_VISIBLE_NORMAL);
    num = 0;
    while (1) {
        cout << "请输入汉诺塔的层数(1-10)" <<
endl;
        cin >> *p_n;
        if (cin.good()) {
            if (*p_n >= 1 && *p_n <= 10) {
                cin.clear();
                while ((getchar()) != '\n');
                break;
            }
            else {
                cin.clear();
                while ((getchar()) != '\n');
                continue;
            }
        }
        else {
            cin.clear();
            while ((*p_n = getchar()) != '\n' &&
*p_n != EOF)
                ;
        }
    }
    while (1) {
        cout << "请输入起始柱(A-C)" << endl;
        cin >> *p_src;
        if (cin.good()) {
            if (*p_src == 'A' || *p_src == 'B' ||
*p_src == 'C' || *p_src == 'a' || *p_src == 'b'
|| *p_src == 'c') {
                cin.clear();
                while ((getchar()) != '\n');
                break;
            }
            else {
```

```
                cin.clear();
                while ((getchar()) != '\n');
                continue;
            }
        }
        else {
            cin.clear();
            while ((*p_src = getchar()) != '\n'
&& *p_src != EOF)
                ;
        }
    }
    if (*p_src == 'A' || *p_src == 'a')
        *p_src = 'A';
    if (*p_src == 'B' || *p_src == 'b')
        *p_src = 'B';
    if (*p_src == 'C' || *p_src == 'c')
        *p_src = 'C';
    while (1) {
        cout << "请输入目标柱(A-C)" << endl;
        cin >> *p_dst;
    //判断输入错误与起始柱类似，略
    //转换*p_dst与*p_src类似，略
    *p_tmp = 'A' + 'B' + 'C' - *p_src - *p_dst;
    if (solution == 4||solution==8) {
        while (1) {
            cout << "请输入移动速度(0-5: 0-按回车
单步演示 1-延时最长 5-延时最短)" << endl;
            cin >> speed;
//判断输入错误与层数类似，略
        }
    }
}
void print_abc_number()
{   int i;
    cout << " ";
    cout << "A:";
    for (i = 0;i < 10;i++) {
        if (s1[i] == 0)
            cout << "  ";
        else
            cout << setw(2) << s1[i];
    }
   //横向输出BC柱上的盘号与A类似，略
    cout << endl;
}
void output_heng(int n, char src, char tmp, char
dst, int solution)
{   int t=0;
    int i = 0;
    if (src == 'A') {
        i = s1[--top1];
        if (solution == 9)
            t = s1[top1];
        s1[top1] = 0;
    }
    //src为B或C的情况与A类似，略
    if (dst == 'A')
        s1[top1++] = i;
    if (dst == 'B')
        s2[top2++] = i;
    if (dst == 'C')
        s3[top3++] = i;
    if (solution == 9)
        cout << "第" << setw(4) << num << " 步("
<< setw(2) << t << "#: " << src << "-->" << dst
<< ") ";
    else
        cout << "第" << setw(4) << num << " 步("
<< setw(2) << n << "#: " << src << "-->" << dst
<< ") ";
    print_abc_number();
}
void output_zong(int n,char src,char tmp,char
dst,int solution,int base_x,int base_y)
{   int x, y, i;
    if (num == 0) {
        cct_gotoxy(base_x -2, base_y -1);
        cout << "=======================";
        cct_gotoxy(base_x, base_y);
        cout << "A         B         C";
    }
    x = base_x;
    y = base_y - 2;
    for (i = 0;i < 10;i++) {
        if (s1[i] == 0) {
            cct_gotoxy(x, y);
            cout << "   ";
            y = y - 1;
        }
        else {
            cct_gotoxy(x, y);
            cout << s1[i] << ' ';
            y = y - 1;
        }
    }
    x = base_x + 10;
    y = base_y - 2;
    //纵向输出B柱上盘号与A类似，略
    x = base_x + 20;
    y = base_y - 2;
    //纵向输出C柱上盘号与A类似，略
    cct_gotoxy(0, base_y + 5);
}
void back_array(char src, char dst)
{   int i = 0;
    if (dst == 'A') {
        i = s1[--top1];
        s1[top1] = 0;
    }
    //dst为B或C的情况与A类似，略
    if (src == 'A')
        s1[top1++] = i;
    //src为B或C的情况与A类似，略
```

```
}
void ahead_array(char src, char dst)
{   int i = 0;
    if (src == 'A') {
        i = s1[--top1];
        s1[top1] = 0;
    }
    //src为B或C的情况与A类似，略
    if (dst == 'A')
        s1[top1++] = i;
}
void draw_moving_plates(int n, char src, char
dst)
{   back_array(src, dst);
    int i, j;
    int x = 12, y;
    cct_setcursor(CURSOR_INVISIBLE);
    if (src == 'A') {
        x = 12;
        for (i = 9;i >= 0;i--) {
            if (s1[i] != 0) {
                for (y = 14 - i;y >= 1;y--) {
                    cct_showch(x - s1[i], y, ' ',
s1[i], s1[i], 2 * s1[i] + 1);
                    if (speed == 0)
                        Sleep(20);
                    else
                        Sleep(10 / speed);
                    if (y >= 3) {
                        cct_showch(x - s1[i], y,
' ', COLOR_BLACK, COLOR_BLACK, s1[i]);
                        cct_showch(x, y, ' ',
COLOR_HYELLOW, COLOR_HYELLOW, 1);
                        cct_showch(x + 1, y, ' ',
COLOR_BLACK, COLOR_BLACK, s1[i]);
                    }
                    if (y == 2)
                        cct_showch(x - s1[i], y,
' ', COLOR_BLACK, COLOR_BLACK, 2 * s1[i] + 1);
                }//上移
                y = 1;
                if (dst == 'B') {
                    for (x = 12;x <= 44;x++) {
                        cct_showch(x - s1[i], y,
' ', s1[i], s1[i], 2 * s1[i] + 1);
                        if (speed == 0)
                            Sleep(20);
                        else
                            Sleep(10 / speed);
                        if (x <= 43)
                            cct_showch(x - s1[i],
y, ' ', COLOR_BLACK, COLOR_BLACK, 2 * s1[i] + 1);
                    }
                    for (j = 9;j >= 0;j--)
                        if (s2[j] != 0)
                            break;
                    x = 44;
                    for (y = 1;y <= 13 - j;y++) {
                        cct_showch(x - s1[i], y,
' ', s1[i], s1[i], 2 * s1[i] + 1);
                        if (speed == 0)
                            Sleep(20);
                        else
                            Sleep(10 / speed);
                        if (y <= 12 - j) {
                            cct_showch(x - s1[i],
y, ' ', COLOR_BLACK, COLOR_BLACK, s1[i]);
                            cct_showch(x, y, ' ',
COLOR_HYELLOW, COLOR_HYELLOW, 1);
                            cct_showch(x + 1, y,
' ', COLOR_BLACK, COLOR_BLACK, s1[i]);
                        }
                        if (y == 1 || y == 2)
                            cct_showch(x - s1[i],
y, ' ', COLOR_BLACK, COLOR_BLACK, 2 * s1[i] + 1);
                    }
                }
//dst为C的情况与B类似，略
                break;
            }
        }
    }
//src为B或C的情况与A类似，略
    ahead_array(src, dst);
}
void hanoi_print_step(int n, char src, char tmp,
char dst, int solution)
{   if(solution==1)
        cout << setw(2) << n << "# " << src << "-
->" << dst << endl;
    if (solution == 2) {
        num = num + 1;
        cout << "第" << setw(4) << num << " 步("
<< setw(2) << n << "# " << src << "-->" << dst <<
")" << endl;
    }
    if (solution == 3) {
        num = num + 1;
        output_heng(n, src, tmp, dst, solution);
    }
    if (solution == 4) {
        num = num + 1;
        if (speed == 0) {
            while (1) {
                if (_getch() == 13) {
                    cct_gotoxy(0, 19);
                    output_heng(n, src, tmp, dst,
solution);
                    output_zong(n, src, tmp, dst,
solution,10,15);
                    break;
                }
                else
                    continue;
```

```
            }
        }
        else {
            Sleep(500 / speed);
            cct_gotoxy(0, 19);
            output_heng(n, src, tmp, dst,
solution);
            output_zong(n, src, tmp, dst,
solution,10,15);
        }
    }
    if (solution == 8) {
        num = num + 1;
        if (speed == 0) {
            while (1) {
                if (_getch() == 13) {
                    cct_setcolor();
                    cct_gotoxy(0, 32);
                    output_heng(n, src, tmp, dst,
solution);
                    output_zong(n, src, tmp, dst,
solution, 11, 28);
                    draw_moving_plates(n, src,
dst);
                    cct_setcolor();
                    break;
                }
                else
                    continue;
            }
        }
        else {
            Sleep(500 / speed);
            cct_setcolor();
            cct_gotoxy(0, 32);
            output_heng(n, src, tmp, dst,
solution);
            output_zong(n, src, tmp, dst,
solution, 11, 28);
            draw_moving_plates(n, src, dst);
            cct_setcolor();
        }
    }
}
void initialize_top(char *p_src, int *p_n)
{   int i, j;
    j = *p_n;
    if (*p_src == 'A') {
        for (i = 0;i <= *p_n - 1;i++) {
            s1[i] = j;
            j = j - 1;
        }
        top1 = *p_n;
    }
    //*p_src为B或C的情况与A类似，略
}
void hanoi(int n, char src, char tmp, char dst,
int solution)
{   if (n > 1) {
        hanoi(n - 1, src, dst, tmp, solution);
        hanoi_print_step(n, src, tmp, dst,
solution);
        hanoi(n - 1, tmp, src, dst, solution);
    }
    if (n == 1) {
        hanoi_print_step(n, src, tmp, dst,
solution);
    }
}
void hanoi_continue()
{   cct_setcursor(CURSOR_VISIBLE_NORMAL);
    cout << "按回车键继续";
    while (1) {
        if (_getch() == 13)
            break;
    }
}
void to_be_continued(const int X = 0, const int Y
= 38)
{
    cct_setcursor(CURSOR_VISIBLE_NORMAL);
    cct_setcolor(); //恢复缺省颜色
    cct_gotoxy(X, Y);
    cout << "按回车键继续";
    while (_getch() != '\r')
        ;
    return;
}
void draw_column()
{   cct_setcursor(CURSOR_INVISIBLE);
    int x = 1, y = 15;
    for (x = 1;x <= 23;x++)
        cct_showch(x, y, ' ', COLOR_HYELLOW,
COLOR_HYELLOW, 1);
//画BC柱子底座与仅画A柱位置不同，略
    for (y = 14;y >= 3;y--)
        for (x = 12;x <= 76;x = x + 32) {
            Sleep(20);
            cct_showch(x, y, ' ', COLOR_HYELLOW,
COLOR_HYELLOW, 1);
        }
    cct_setcolor();
}
void draw_plates()
{   int x, y, i, j;
    x = 12, y = 14;
    cct_setcursor(CURSOR_INVISIBLE);
    for (i = 0;i < 10;i++) {
        if (s1[i] == 0) {
            for (j = 1;j <= 10;j++)
                cct_showch(x - 11 + j, y, ' ',
COLOR_BLACK, COLOR_BLACK, 1);
            cct_showch(x, y, ' ', COLOR_HYELLOW,
COLOR_HYELLOW, 1);
```

```
            for (j = 1;j <= 10;j++)
                cct_showch(x + j, y, ' ',
COLOR_BLACK, COLOR_BLACK, 1);
        }
        else {
            Sleep(20);
            for (j = 1;j <= 2 * s1[i] + 1;j++)
                cct_showch(x - s1[i] - 1 + j, y,
' ', s1[i], s1[i], 1);
        }
        y = y - 1;
    }
    x = 44, y = 14;
    //画B柱上盘子与A类似，略
    x = 76, y = 14;
    //画C柱上盘子与A类似，略
    cct_setcolor();
}
void reset_array()
{   int i;
    for (i = 0;i < 10;i++)
        s1[i] = 0;
    for (i = 0;i < 10;i++)
        s2[i] = 0;
    for (i = 0;i < 10;i++)
        s3[i] = 0;
    top1 = 0, top2 = 0, top3 = 0;
}
int judge_yuanzhu(char letter1)
{   int judge = 0, i;
    if (letter1 == 'A') {
        for (i = 0;i < 10;i++) {
            if (s1[i] != 0) {
                judge = 1;
                break;
            }
            judge = 0;
        }
    }
    //letter1为B或C的情况与上述相似，略
    return judge;
}
int judge_big_above_small(char letter1, char
letter2)
{   int judge = 0, plate1 = 0, plate2 = 0,i;
    if (letter1 == 'A') {
        for (i = 9;i >= 0;i--) {
            if (s1[i] != 0) {
                plate1 = s1[i];
                break;
            }
        }
    }
    //letter1为B或C的情况与上述相似，略
    if (letter2 == 'A') {
        for (i = 9;i >= 0;i--) {
            if (s1[i] != 0) {
                plate2 = s1[i];
                break;
            }
        }
    }
    //letter1为B或C的情况与上述相似，略
    if (plate2 == 0) {
        judge = 1;
    }
    else {
        if (plate1 > plate2)
            judge = 0;
        else
            judge = 1;
    }
    return judge;
}
int judge_success(int n, char src, char tmp, char
dst, char m,int k)
{   int i, judge = 1;
    if (m == 'A') {
        for (i = 0;i < k;i++) {
            if (s1[i] != k - i) {
                judge = 0;
                break;
            }
        }
    }
    //m为B或C的情况与上述相似，略
    return judge;
}
void move_solution9(int n, char src, char tmp,
char dst,char m,int k)
{   int i;
    char x = 0;
    char letter[128] = { 0 };
    //char letter1, letter2;
    cct_gotoxy(0, 34);
    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) ：";
    while (1) {
        for (i = 0;i < 127;i++)
            letter[i] = ' ';
        letter[127] = 0;
        cct_gotoxy(61, 34);
        cout << letter;
        cct_gotoxy(61, 34);
        cct_setcursor(CURSOR_VISIBLE_NORMAL);
        for (i = 0;i < 127;i++) {
            x = _getch();
            if (x >= 33 && x <= 126) {
                letter[i] = x;
                cout << letter[i];
            }
            if (x == 13)
                break;
        }
```

```
        letter[i] = 0;
        if (strlen(letter) == 1) {
            if (letter[0] == 'q' || letter[0] == 'Q') {
                cct_gotoxy(0, 35);
                cout << "游戏中止!!!!!" << endl;
                break;
            }
        }
        else if (strlen(letter) == 2) {
            if ((letter[0] == 'a' || letter[0] == 'A' || letter[0] == 'b' || letter[0] == 'B' || letter[0] == 'c' || letter[0] == 'C')&& (letter[1] == 'a' || letter[1] == 'A' || letter[1] == 'b' || letter[1] == 'B' || letter[1] == 'c' || letter[1] == 'C')) {
                if (letter[0] == 'a' || letter[0] == 'b' || letter[0] == 'c')
                    letter[0] = letter[0] - 'a' + 'A';
                if (letter[1] == 'a' || letter[1] == 'b' || letter[1] == 'c')
                    letter[1] = letter[1] - 'a' + 'A';
                if (judge_yuanzhu(letter[0]) == 0) {
                    cct_gotoxy(0, 35);
                    cout << "源柱为空!";
                    Sleep(1000);
                    cct_gotoxy(0, 35);
                    cout << "          ";
                    continue;
                }
                if (judge_big_above_small(letter[0], letter[1]) == 0) {
                    cct_gotoxy(0, 35);
                    cout << "大盘压小盘，非法移动!";
                    Sleep(1000);
                    cct_gotoxy(0, 35);
                    cout << "
";
                    continue;
                }
                src = letter[0];
                dst = letter[1];
                tmp = 'A' + 'B' + 'C' - letter[0] - letter[1];
                num = num + 1;
                cct_setcolor();
                cct_gotoxy(0, 32);
                output_heng(n, src, tmp, dst, 9);
                output_zong(n, src, tmp, dst, 9, 11, 28);
                draw_moving_plates(n, src, dst);
                cct_setcolor();
                if (judge_success(n, src, tmp, dst, m, k)) {
                    cct_gotoxy(0, 35);
                    cout << "游戏结束!!!!!";
                    break;
                }
            }
        }
        else
            continue;
    }
}
void solution0()
{   cct_gotoxy(0, 38);
}
void solution1()
{   int n = 0, speed = 0;
    char src = 0, tmp = 0, dst = 0;
    hanoi_input(&n, &src, &tmp, &dst, 1);
    hanoi(n, src, tmp, dst, 1);
    cout << endl;
    hanoi_continue();
}
//void solution2()仅部分参数与solution1不同，略
void solution3()
{   int n = 0, speed = 0;
    char src = 0, tmp = 0, dst = 0;
    reset_array();
    hanoi_input(&n, &src, &tmp, &dst,3);
    initialize_top(&src, &n);
    hanoi(n, src, tmp, dst, 3);
    cout << endl;
    hanoi_continue();
}
void solution4()
{   int n = 0,speed = 0,k;
    char src = 0, tmp = 0, dst = 0;
    reset_array();
    hanoi_input(&n, &src, &tmp, &dst, 4);
    k = n;
    initialize_top(&src, &n);
    cct_setcursor(CURSOR_INVISIBLE);
    cct_cls();
    cct_gotoxy(0, 0);
    cout << "从 " << src << " 移动到 " << dst << "，共 " << k << " 层，延时设置为 " << speed;
    output_zong(n, src, tmp, dst, 4, 10, 15);
    cct_gotoxy(0, 19);
    cout << "初始:                ";
    print_abc_number();
    hanoi(n, src, tmp, dst, 4);
    cct_gotoxy(0, 38);
    hanoi_continue();
}
void solution5()
{   cct_cls();
    cct_setcursor(CURSOR_INVISIBLE);
```

```
    draw_column();
    cct_setcolor();
    cct_gotoxy(0, 38);
    to_be_continued();
}
void solution6()
{   int n = 0, speed = 0, k;
    char src = 0, tmp = 0, dst = 0;
    reset_array();
    hanoi_input(&n, &src, &tmp, &dst, 6);
    initialize_top(&src, &n);
    k = n;
    cct_cls();
    cct_setcursor(CURSOR_INVISIBLE);
    cout << "从 " << src << " 移动到 " << dst <<
", 共 " << k << " 层";
    draw_column();
    draw_plates();
    cct_gotoxy(0, 38);
    to_be_continued();
}
void solution7()
{   int n = 0, k;
    char src = 0, tmp = 0, dst = 0;
    speed = 0;
    reset_array();
    hanoi_input(&n, &src, &tmp, &dst, 6);
    initialize_top(&src, &n);
    k = n;
    cct_cls();
    cct_setcursor(CURSOR_INVISIBLE);
    cout << "从 " << src << " 移动到 " << dst <<
", 共 " << k << " 层";
    draw_column();
    draw_plates();
    if (k % 2 == 1)
        draw_moving_plates(1, src, dst);
    else
        draw_moving_plates(1, src, tmp);
    cct_gotoxy(0, 38);
    to_be_continued();
}
void solution8()
{   int n = 0,speed = 0,k;
    char src = 0, tmp = 0, dst = 0;
    reset_array();
    hanoi_input(&n, &src, &tmp, &dst, 8);
    k = n;
    initialize_top(&src, &n);
    cct_cls();
    cct_setcursor(CURSOR_INVISIBLE);
    cct_gotoxy(0, 0);
    cout << "从 " << src << " 移动到 " << dst <<
", 共 " << k << " 层, 延时设置为 " << speed;
    output_zong(n, src, tmp, dst, 8, 11, 28);
    cct_gotoxy(0, 32);
    cout << "初始:               ";
    print_abc_number();
    draw_column();
    draw_plates();
    hanoi(n, src, tmp, dst, 8);
    cct_gotoxy(0, 38);
    to_be_continued();
}
void solution9()
{   int n = 0,speed = 0,k;
    char m = 0,src = 0, tmp = 0, dst = 0;
    reset_array();
    hanoi_input(&n, &src, &tmp, &dst, 9);
    k = n;
    m = dst;
    initialize_top(&src, &n);
    cct_cls();
    cct_setcursor(CURSOR_INVISIBLE);
    cct_gotoxy(0, 0);
    cout << "从 " << src << " 移动到 " << dst <<
", 共 " << k << " 层, 延时设置为 " << speed;
    output_zong(n, src, tmp, dst, 8, 11, 28);
    cct_gotoxy(0, 32);
    cout << "初始:               ";
    print_abc_number();
    draw_column();
    draw_plates();
    while (1) {
        if (_getch() == 13) {
            move_solution9(n, src, tmp, dst, m,
k);
            break;
        }
    }
    cct_gotoxy(0, 38);
    to_be_continued();
}
```