汉诺塔综合演示实验报告

班级：软件5班

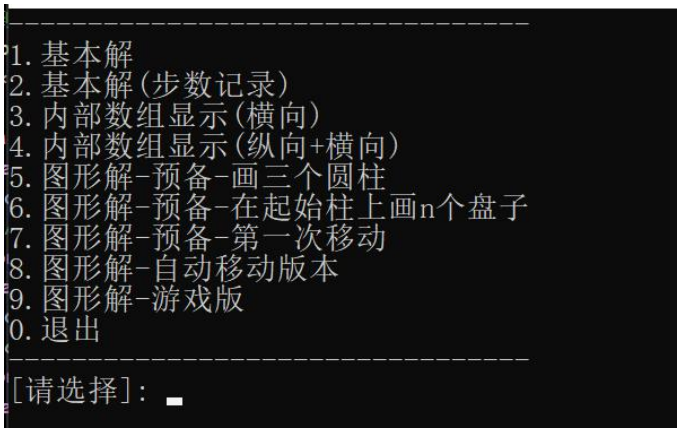
学号：2152402

姓名：段婷婷

完成日期：2022.11.24

1. 题目及基本要求描述

实现汉诺塔综合演示，用菜单方式进行选择，实现以下选项功能：



选项 1：输出汉诺塔的解的移动步骤
选项 2：在选项1的基础上，记录步数并输出
选项 3：在选项2的基础上，记录每一个柱上的盘的信息并横向输出
选项 4：在选项3的基础上，还要纵向地输出每一个柱上盘的信息
选项 5：在屏幕上画出三根圆柱
选项 6：在选项5的基础上，在起始圆柱上从小到大画出n个圆盘，每个圆盘的颜色各不相同
选项 7：在选项6的基础上，完成第一个圆盘的移动（每次圆盘的移动方式必须是上移、平移、下移）
选项 8：汉诺塔演示过程的完整实现（每次圆盘的移动方式必须是上移、平移、下移）
选项 9：汉诺塔游戏（人工操作移动步骤）

2. 整体设计思路

2.1. 各个源文件名以及作用

整个程序由6个源文件组成：cmd_console_tools.h，hanoi.h，cmd_console_tools.cpp，hanoi_main.cpp，hanoi_menu.cpp，hanoi_multiple_solutions.cpp。

其中 cmd_console_tools.h 和 cmd_console_tools.cpp分别是给定的伪图形界面下基本功能函数的函数声明和具体实现。

hanoi.h 用于声明 hanoi_menu.cpp 和 hanoi_multiple_solutions.cpp中需要用到的函数。

hanoi_menu.cpp 用于实现菜单的显示以及读入正确的选项后返回的函数（即：menu()）。

hanoi_multiple_solutions.cpp 用于实现菜单项中各个选项功能的函数。

hanoi_main.cpp 用于调用menu()函数并且根据返回值调用hanoi_multiple_solutions.cpp中对应的

执行函数。

2.2. hanoi_multiple_solutions.cpp 中执行函数的设计思路

主要分为以下五类：

1. main函数中根据menu()返回值直接调用的函数。
2. 显示横向、纵向数组的函数。
3. 显示汉诺塔图像（柱、盘、盘的移动）的函数。
4. 求解汉诺塔问题的递归函数及其输出函数。
5. 辅助以上函数工作的函数。

以上函数的具体介绍见下一个部分【3.主要功能的实现】

3. 主要功能的实现

3.1. 各个函数的功能

3.1.1. hanoi_menu.cpp中的菜单函数：

函数声明：int menu();
函数功能：显示菜单，读入正确的选项后返回

3.1.2. hanoi_multiple_solutions.cpp中main()直接调用的函数

```
（1）void sol_for_123(int opt); //当选择菜单项1~3时调用
（2）void sol_for_opt4();       //当选择菜单项4时调用
（3）void draw_for_opt5();      //当选择菜单项5时调用
（4）void draw_for_opt6();      //当选择菜单项6时调用
（5）void draw_for_opt7();      //当选择菜单项7时调用
（6）void draw_for_opt8();      //当选择菜单项8时调用
（7）void draw_for_opt9();      //当选择菜单项9时调用
```

3.1.3. hanoi_multiple_solutions.cpp中显示横向、纵向数组的函数

```
（1）void show_array_transverse(); //用于显示横向数组
（2）void show_array_portrait(int Y); //用于显示纵向数组，参数Y控制显示位置
（3）void show_arrays(char src, char dst, int x, int y)
   //在初始或移动完一步之后更新显示横向和纵向数组
```

3.1.4. hanoi_multiple_solutions.cpp中显示汉诺塔图像（柱、盘、盘的移动）的函数

```
(1) void draw_cylinder();                    //画三个圆柱
```

(2) void draw_src_plates(char src, int n); //画起始柱src上的n个盘子

(3) void init_graph(int n, char src, char dst, bool show_delay);

//初始化整个汉诺塔图形界面,包括: 圆柱、起始柱上的n个盘子、页面顶部显示的提示语句（从*移动到*，共*层，延时设置为*）

(4) void draw_move(int n, char src, char dst);

实现功能：画出将n圆盘从src柱移到dst柱的全过程。

主要步骤：第n个盘子在src柱中纵向上移，盘从src柱横向移到dst盘，盘在dst盘中纵向下移。

3.1.5.hanoi_multiple_solutions.cpp中求解汉诺塔问题的递归函数及其输出函数

（1）void hanoi(int n, char src, char tmp, char dst, int opt);

函数功能：通过递归求解汉诺塔问题

参数含义：当前需要将总共n个盘子从src柱移动到dst柱，opt表示选项，以此来决定输出移动步骤的方式。

具体实现：

```
void hanoi(int n, char src, char tmp, char dst, int opt)
{
    if (n == 1)
    {
        hanoi_output_move(n, src, dst, opt);
        return;
    }
    hanoi(n - 1, src, dst, tmp, opt);
    hanoi_output_move(n, src, dst, opt);
    hanoi(n - 1, tmp, src, dst, opt);
}
```

（2）void hanoi_output_move(int n, char src, char dst, int opt);

函数功能：在hanoi函数中对盘子移动时，以opt对应的方式显示移动步骤。

参数含义：当前需要将第n个盘子从src柱移动到dst柱，opt表示选项，以此来决定输出移动步骤的方式。

3.1.6. hanoi_multiple_solutions.cpp中辅助以上函数工作的函数

（1）void init_input(int& n, char& src, char& dst, char& tmp, bool delay);

// 输入n,src,dst,delay_time与初始化tmp,top[],stack[][].

（2）void Pause(int y); //实现在菜单项功能完成后暂停，按回车继续的功能

（3）void go_to_next_step(bool enter);

函数功能：在设置了延时的情况下，用于实现不同延时的功能

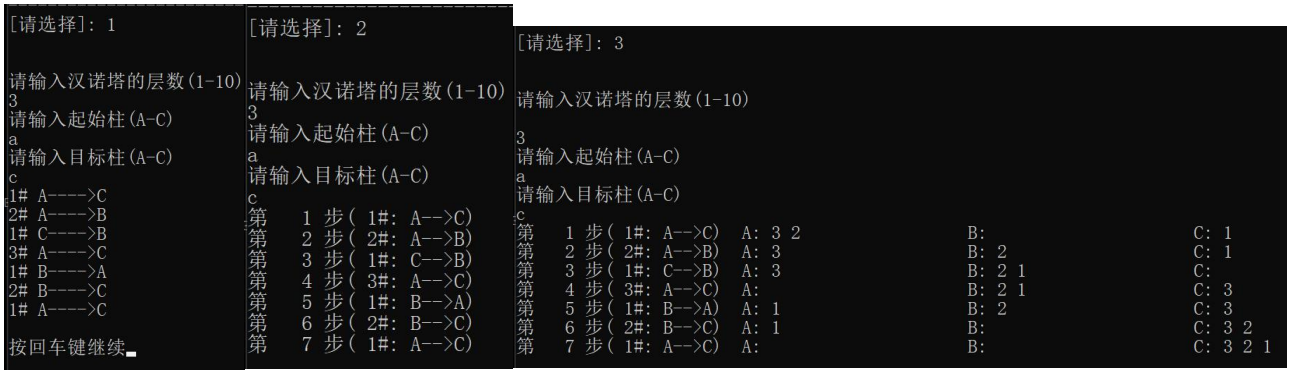
参数含义：enter用于判断静态全局变量delay_time=0时，含义是延时最短还是按回车进行下一步。

3.2 各个菜单项的实现：

3.2.1. 基本解/基本解（步数记录）/内部数组显示（横向）

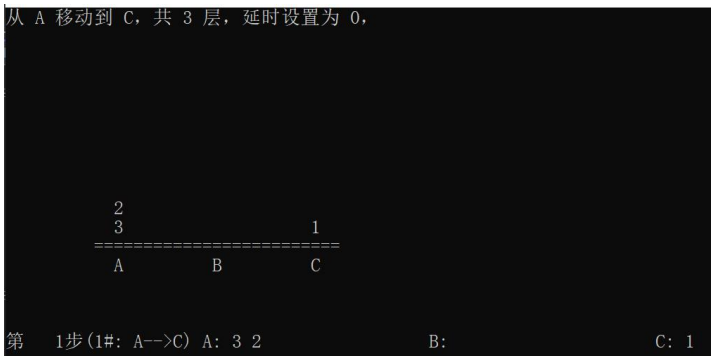
由main()调用sol_for_opt123(int),在sol_for_opt123(int)中主要分为两步：调用init_input函数

来输入和初始化，调用hanoi函数来解汉诺塔问题并且输出步骤。功能示例如下：



3.2.2. 内部数组显示(纵向+横向)

由main()调用sol_for_opt4()，在sol_for_opt4()中主要分为三步：调用init_input函数来输入和初始化变量，调用show_arrays函数来显示初始的横向与纵向数组，调用hanoi函数来解汉诺塔问题并且输出步骤。功能示例如下：



3.2.3. 图形解-预备-画三个圆柱

由main()调用draw_for_opt5()，在draw_for_opt5()中调用draw_cylinder函数即可。

3.2.4. 图形解-预备-在起始柱上画n个盘子

由main()调用draw_for_opt6()，在draw_for_opt6()中主要分为两步：调用init_input函数来输入和初始化变量，调用init_graph函数来画出初始的汉诺塔图形显示界面（即：圆柱与起始柱上的n各圆盘）。

3.2.5. 图形解-预备-第一次移动

由main()调用draw_for_opt7()，在draw_for_opt7()中主要分为四步：调用init_input函数来输入和初始化变量，调用init_graph函数来画出初始的汉诺塔图形显示界面，在数组中移动第一步，调用draw_move函数来画出盘移动的过程。

如何判断第一步如何移动？换言之，第1个盘是移至dst还是tmp？易得，当n为奇数时，移至dst；当n为偶数时，移至tmp。

3.2.6. 图形解-自动移动版本

由main()调用draw_for_opt8()，在draw_for_opt8()中主要分为四步：调用init_input函数来输入和初始化变量，调用init_graph函数来画出初始的汉诺塔图形显示界面，调用show_arrays函数来显示初始的横向与纵向数组，调用hanoi函数来解汉诺塔问题并且输出移动步骤。

3.2.7. 图形解-游戏版

由main()调用draw_for_opt9()，在draw_for_opt9()中主要分为四步：调用init_input函数来输入和初始化变量，调用init_graph函数来画出初始的汉诺塔图形显示界面，调用show_arrays函数来显示初始的横向与纵向数组，读入正确的移动指令并且输出移动过程直至完成所有移动或读到退出指令。

4. 调试过程碰到的问题

在调试过程中碰到了很多问题，其中有许多可以归结为与多处调用同一函数相关的两类问题，现选取有代表性的两个问题阐述：

4.1.

(1)**问题**：调用draw_move来输出移动过程时，在sol_for_opt8中正确但在sol_for_opt7中错误。
(2)**差错方法与错处**：检查在sol_for_opt7和sol_for_opt8中调用draw_move前，draw_move中需要用到的静态全局变量的状态是否相同。发现在sol_for_opt8中，调用draw_move前，已经根据移动信息改变了top[]，stack[][]，使二者处于移动完成的状态；而在sol_for_opt7中，top[]和stack[][]处于移动前的状态。这个差异使得draw_move函数表现不同。
(3)**解决方法**：在sol_for_opt7调用draw_move前，根据移动信息改变top[]与stack[][]，使二者处于移动完成的状态。
(4)**小结**：在不同的地方调用同一个函数时，注意统一处理方式。

4.2.

(1)**问题**：调用show_arrays来输出横向与纵向数组时，在sol_for_opt4中显示正确，但在sol_for_opt8中显示位置不合适。
(2)**解决方法**：给show_arrays函数加一个参数，用来控制输出数组的位置。
(3)**小结**：若函数不能满足所有调用处的需求，可以考虑添加参数来差异化地解决主体相同但细节处理不通过地问题。

5. 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

在写这种有很多项的程序时，可以先分别写出每一项的实现，然后对其中重复的、相似的部分进行精简整合。在写代码的过程中最最重要的是理清逻辑思路，可以事半功倍。

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？

分成若干小题的形式更好。

一方面，将复杂大题分成若干小题符合思考的逻辑，对于任何复杂的问题都应该将其划分成若干简化的小问题，依次解决了简化的小问题也就解决了复杂的问题；另一方面，我们应该学会整合部分的程序来形成完整的程序，因为在后续更深入的学习中，可能会遇到多人分工写程序的情况，此时写好部分程序与整合程序都是必要的。

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？

在完成汉诺塔小题的过程中，我考虑了前后小题的关联关系，并且尽可能做到后面小题有效利用前面小题已完成的代码。

如何更好地重用代码？

①在写代码时就要对代码有清晰的划分，明确每个部分实现的功能，方便后面需要相似的功能时可以快速地找到之前写过的部分，重用代码。

②将一部分实现某一特定功能的代码写成函数有利于重用代码。

③对于函数，尽量做到函数中用到的变量都是参数表中的或函数内定义的，尽量少用全局变量，这样重用代码时只需要移植函数部分，在其内部进行需要的改动即可。

④对于同样意义的变量统一命名，并且命名要能清楚地读出其含义，这样可以在重用时减少改动的部分，便于重用代码。

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

①通过函数来分割代码。对整个代码进行清晰的划分，明确每个部分实现的功能，并且将不同的部分写成单独的函数。这样使得复杂的程序变得一目了然、逻辑清晰。

②将多处出现的实现相似功能的代码统一成一个函数，能够有效减少冗余、精简代码。在写代码前就意识到这点的话可以有效提高编写程序的效率和正确性。

6. 附件：源程序

（1）hanoi.h

```
/*2152402 软件 段婷婷*/
#pragma once
using namespace std;
int menu();

void sol_for_123(int opt);
void sol_for_opt4();
void draw_for_opt5();
void draw_for_opt6();
void draw_for_opt7();
void draw_for_opt8();
void draw_for_opt9();

void init_input(int& n, char& src, char& dst, char& tmp,
bool delay);
void Pause(int y);
void go_to_next_step(bool enter);

void show_array_transverse();
void show_array_portrait(int Y);
void show_arrays(char src, char dst, int x, int y);
void draw_cylinder();
void draw_src_plates(char src, int n);
void init_graph(int n, char src, char dst, bool
show_delay);
void draw_move(int n, char src, char dst);

void hanoi_output_move(int n, char src, char dst, int
opt);
void hanoi(int n, char src, char tmp, char dst, int
opt);
```

(2) hanoi_menu.cpp

```
/*2152402 软件 段婷婷*/
#include "hanoi.h"
#include "cmd_console_tools.h"
#include <iostream>
#include <conio.h>
using namespace std;

int menu()
{
    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]: ";
    char c = _getch();
    while (c < '0' || c > '9')
        c = _getch();
    return c - '0';
}
```

(3) hanoi_multiple_solutions.cpp

```
/*2152402 软件 段婷婷*/
#include "cmd_console_tools.h"
#include "hanoi.h"
#include <iostream>
#include <iomanip>
#include <conio.h>
#include<windows.h>
using namespace std;

static int steps = 0;
static int top[3], stack[3][10];
static int delay_time;

void sol_for_123(int opt)
{
    int n;
    char src, dst, tmp;
    init_input(n, src, dst, tmp, 0);

    hanoi(n, src, tmp, dst, opt);

    Pause(-1);
}

void sol_for_opt4()
{
    int n;
    char src, dst, tmp;
    init_input(n, src, dst, tmp, 1);

    cct_cls();
    cout << "从 " << src << " 移动到 " << dst << ",
" << "共 " << n << " 层, ";
    cout << "延时设置为 " << delay_time << ", ";
    show_arrays(src, dst, 0, 17);

    hanoi(n, src, tmp, dst, 4);

    Pause(27);
}
void draw_for_opt5()
```

```
{
    cct_cls();
    draw_cylinder();

    Pause(27);
}
void draw_for_opt6()
{
    int n;
    char src, dst, tmp;
    init_input(n, src, dst, tmp, 0);
    init_graph(n, src, dst, 0);

    Pause(27);
}
void draw_for_opt7()
{
    int n;
    char src, dst, tmp;
    init_input(n, src, dst, tmp, 0);
    init_graph(n, src, dst, 0);

    char to = n % 2 == 1 ? dst : tmp;
    stack[to - 'A'][top[to - 'A']++] = stack[src -
'A'][--top[src - 'A']];
    draw_move(1, src, to);

    Pause(27);
}
void draw_for_opt8()
{
    int n;
    char src, dst, tmp;
    init_input(n, src, dst, tmp, 1);
    init_graph(n, src, dst, 1);

    show_arrays(src, dst, 0, 32);
    hanoi(n, src, tmp, dst, 8);

    Pause(37);
}
void draw_for_opt9()
{
    int n;
    char src, dst, tmp;
    init_input(n, src, dst, tmp, 0);
    delay_time = 2;

    init_graph(n, src, dst, 0);
    show_arrays(src, dst, 0, 32);

    while (1)
    {
        cct_gotoxy(0, 34);
        cct_setcursor(2);
        cct_setcolor();
        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)：        ";
        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)：";

        char fr = _getche(), to = _getche();
        if (fr == 'q' || fr == 'Q')
        {
            cct_gotoxy(0, 35);
            cout << "游戏中止!!!!!";
            break;
        }
        if (fr > 'C')
            fr -= 32;
        if (to > 'C')
            to -= 32;
        if (fr < 'A' || fr>'C' || to < 'A' || to>'C'
|| fr == to)
            continue;
        if (top[fr - 'A'] == 0)
        {
            cct_gotoxy(0, 35);
            cout << "柱源为空！";
            Sleep(100);
            cct_gotoxy(0, 35);
            cout << "                              ";
            continue;
        }
        if (top[to - 'A'] && stack[to - 'A'][top[to -
'A'] - 1] < stack[fr - 'A'][top[fr - 'A'] - 1])
        {
            cct_gotoxy(0, 35);
            cout << "非法移动，大盘压小盘！";
            Sleep(100);
            cct_gotoxy(0, 35);
            cout << "                              ";
            continue;
        }

        int x = stack[fr - 'A'][--top[fr - 'A']];
        stack[to - 'A'][top[to - 'A']++] = x;

        show_arrays(fr, to, x, 32);
        draw_move(x, fr, to);

        if (top[dst - 'A'] == n)
        {
            cct_gotoxy(0, 35);
            cct_setcolor();
            cout << "游戏中止!!!!!";
            break;
        }
    }

    Pause(38);
}
void init_input(int& n, char& src, char& dst, char& tmp,
```

```
bool delay)
{
    steps = 0;
    top[0] = top[1] = top[2] = 0;
    while (1)
    {
        cout << "请输入汉诺塔的层数(1-10)" << endl;
        cin >> n;
        if (!cin.good())
        {
            cin.clear();
            cin.ignore(INT_MAX, '\n');
        }
        if (n >= 1 && n <= 16)
            break;
    }
    while (1)
    {
        cout << "请输入起始柱(A-C)" << endl;
        cin >> src;
        cin.clear();
        cin.ignore(INT_MAX, '\n');
        if (src > 'Z')
            src -= 32;
        if (src >= 'A' && src <= 'C')
            break;
    }
    while (1)
    {
        cout << "请输入目标柱(A-C)" << endl;
        cin >> dst;
        cin.clear();
        cin.ignore(INT_MAX, '\n');
        if (dst > 'Z')
            dst -= 32;
        if (dst == src)
            cout << "目标柱(" << dst << ")不能与起始
柱(" << src << ")相同" << endl;
        if (dst >= 'A' && dst <= 'C' && dst != src)
            break;
    }
    tmp = 'A' + 'C' + 'B' - src - dst;
    for (int i = n;i >= 1;--i)
    {
        int ind = src - 'A';
        stack[ind][top[ind]++] = i;
    }
    if (delay)
    {
        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 >> delay_time;
            if (!cin.good())
            {
                cin.clear();
                cin.ignore(INT_MAX, '\n');
                continue;
            }
            if (delay_time >= 0 && delay_time <= 5)
                break;
        }
    }
    else
        delay = 0;
}
void Pause(int y)
{
    if (y != -1)
        cct_gotoxy(0, y);
    cct_setcolor();
    cout << endl << "按回车键继续";
    int X = 0, Y = 0, ret, maction, keycode1, keycode2;
    while (1)
    {
        ret = cct_read_keyboard_and_mouse(X, Y,
maction, keycode1, keycode2);
        if (ret == CCT_KEYBOARD_EVENT && keycode1 ==
13)
            break;
    }
}
void go_to_next_step(bool enter)
{
    double delay_time_arr[5] =
{ 100,50,1,1e-3,1e-6 };
    if (delay_time == 0 && enter)
    {
        while (1)
        {
            int op2 = _getch();
            if (op2 == 13)
                break;
        }
    }
    else
        Sleep(delay_time_arr[delay_time]);
}
void show_array_transverse()
{
    for (int ind = 0;ind < 3;++ind)
    {
        cout << setw(2) << char(ind + 'A') << ":";
        for (int i = 0;i < top[ind];++i)
            cout << setw(2) << stack[ind][i];
        for (int i = 1;i <= 10 - top[ind];++i)
            cout << "  ";
    }
    cout << endl;
}
void show_array_portrait(int Y)
{
    cct_setcolor();
```

```
    cct_gotoxy(9, Y);
    cout << "========================";
    for (int ind = 0;ind < 3;++ind)
    {
        cct_gotoxy(11 + 10 * ind, Y + 1);
        cout << char('A' + ind);
        int x = 10 + 10 * ind, y = Y;
        for (int i = 0;i < top[ind];++i)
        {
            cct_gotoxy(x, --y);
            cout << setw(2) << stack[ind][i] << " ";
        }
        int rest = top[0] + top[1] + top[2] - top[ind];
        for (int i = 1;i <= rest;++i)
        {
            cct_gotoxy(x, --y);
            cout << "  ";
        }
    }
}
void show_arrays(char src, char dst, int x, int y)
{
    cct_setcolor();
    if (steps != 0)
        go_to_next_step(1);
    cct_gotoxy(0, y);
    if (steps == 0)
        cout << "初始:              ";
    else
        cout << "第" << setw(4) << steps << "步(" << x << "#: " << src << "-->" << dst << ")";
    show_array_transverse();
    show_array_portrait(y - 5);
    ++steps;
}void draw_cylinder()
{
    cct_setcursor(CURSOR_INVISIBLE);
    cct_showch(1, 15, ' ', 14, 0, 23);
    cct_showch(34, 15, ' ', 14, 0, 23);
    cct_showch(67, 15, ' ', 14, 0, 23);

    for (int j = 0;j < 12;++j)
    {
        for (int i = 0;i < 3;++i)
        {
            int x = 12 + i * 33, y = 14;
            cct_showch(x, y - j, ' ', 14, 0, 1);
            Sleep(100);
        }
    }
}
void draw_src_plates(char src, int n)
{
    cct_setcursor(CURSOR_INVISIBLE);
    int x = 12 + 33 * (src - 'A'), y = 14;
    int last_color = -1;
    for (int i = n;i >= 1;--i)
    {
        int bg_color = i;
        cct_showch(x - i, y, ' ', bg_color, 0, i * 2 + 1);
        Sleep(50);
        --y;
    }
}
void init_graph(int n, char src, char dst, bool show_delay)
{
    cct_cls();
    cct_gotoxy(0, 0);
    cout << "从 " << src << " 移动到 " << dst << ", 共 " << n << " 层";
    if (show_delay)
        cout << "延时设置为 " << delay_time;
    draw_cylinder();
    draw_src_plates(src, n);
}
void draw_move(int n, char src, char dst)
{
    int sy = 14 - top[src - 'A'], ty = 15 - top[dst - 'A'];
    int sx = 12 + 33 * (src - 'A'), tx = 12 + 33 * (dst - 'A');
    int bg_color = n;

    for (int y = sy;y >= 1;--y)
    {
        cct_showch(sx - n, y, ' ', bg_color, 0, n * 2 + 1);
        go_to_next_step(0);
        cct_showch(sx - n, y, ' ', 0, 0, n * 2 + 1);
        if (y > 2)
            cct_showch(sx, y, ' ', 14, 0, 1);
    }

    if (sx < tx)
        for (int x = sx;x <= tx;++x)
        {
            cct_showch(x - n, 1, ' ', bg_color, 0, n * 2 + 1);
            go_to_next_step(0);
            cct_showch(x - n, 1, ' ', 0, 0, n * 2 + 1);
        }
    else
        for (int x = sx;x >= tx;--x)
        {
            cct_showch(x - n, 1, ' ', bg_color, 0, n * 2 + 1);
            go_to_next_step(0);
            cct_showch(x - n, 1, ' ', 0, 0, n * 2 + 1);
        }

    for (int y = 1;y <= ty;++y)
    {
```

```
        cct_showch(tx - n, y, ' ', bg_color, 0, n * 2
+ 1);
        go_to_next_step(0);
        if (y == ty)
            break;
        cct_showch(tx - n, y, ' ', 0, 0, n * 2 + 1);
        if (y > 2)
            cct_showch(tx, y, ' ', 14, 0, 1);
    }
}
void hanoi_output_move(int n, char src, char dst, int
opt)
{
    int f_ind = src - 'A', t_ind = dst - 'A';
    if (opt == 1)
        cout << n << "# " << src << "---->" << dst <<
endl;
    else if (opt == 2)
        cout << "第" << setw(4) << ++steps << " 步" <<
"(" << setw(2) << n << "#: " << src << "-->" << dst <<
")" << endl;
    else if (opt == 3)
        cout << "第" << setw(4) << ++steps << " 步(" <<
setw(2) << stack[f_ind][top[f_ind] - 1] << "#: " <<
src << "-->" << dst << ") ";

    int x = stack[f_ind][top[f_ind] - 1];
    stack[t_ind][top[t_ind]++] =
stack[f_ind][--top[f_ind]];

    if (opt == 3)
        show_array_transverse();
    else if (opt == 4)
        show_arrays(src, dst, x, 17);
    else if (opt == 8)
    {
        show_arrays(src, dst, x, 32);
        draw_move(n, src, dst);
    }
}
void hanoi(int n, char src, char tmp, char dst, int opt)
{
    int f_ind = src - 'A', t_ind = dst - 'A';
    if (n == 1)
    {
        hanoi_output_move(n, src, dst, opt);
        return;
    }
    hanoi(n - 1, src, dst, tmp, opt);
    hanoi_output_move(n, src, dst, opt);
    hanoi(n - 1, tmp, src, dst, opt);
}
```

(4) hanoi_main.cpp

```
/*2152402 软件 段婷婷*/
#include "cmd_console_tools.h"
#include "hanoi.h"
#include <iostream>
using namespace std;
int main()
{
    cct_setconsoleborder(120, 40, 120, 9000);
    while (1)
    {
        int opt = menu();
        if (opt == 0)
            break;
        if (opt >= 1 && opt <= 9)
            cout << opt << endl << endl << endl;
        if (opt >= 1 && opt <= 3)
            sol_for_123(opt);
        else if (opt == 4)
            sol_for_opt4();
        else if (opt == 5)
            draw_for_opt5();
        else if (opt == 6)
            draw_for_opt6();
        else if (opt == 7)
            draw_for_opt7();
        else if (opt == 8)
            draw_for_opt8();
        else if(opt==9)
            draw_for_opt9();
    }
    return 0;
}
```