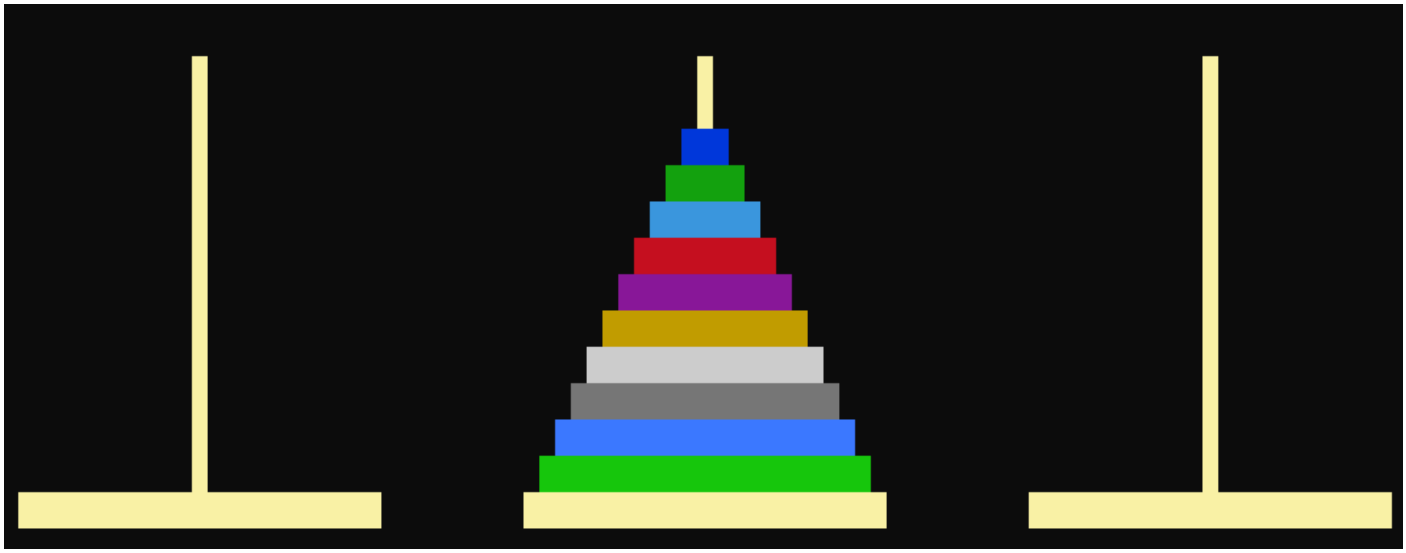
汉诺塔综合演示实验报告



姓　　名：崔露文
班　　级：信03
学　　号：2353126
完成日期：2024年5月14日

1. 题目

1.1.题目总体概述

整合之前做过的所有汉诺塔小题于一个程序中，用菜单方式进行模式的选择（文字解or加入图形解）。

1.2.各菜单功能

菜单一：基本解（对应前序作业4-b12）。
菜单二：基本解+步数记录（对应前序作业4-b13）。
菜单三：内部数组显示（横向）（对应前序作业5-b6）。
菜单四：内部数组显示（纵向+横向）（对应前序作业5-b7）。
菜单五：预备图形解，画出三个基座+柱子。
菜单六：预备图形解，在起始柱上画出n个盘子，即初始化状态。
菜单七：预备图形解，实现第一次移动。
菜单八：实现汉诺塔图形的自动移动。
菜单九：实现汉诺塔的游戏效果，即人工操作移动步骤并判断合理性。

1.3.详细要求与限制

本次作业共6个文件，其中两个头文件：
① cmd_console_tools.h：伪图形界面下基本功能函数的声明。（不提交）
② hanoi.h：本项目的头文件，为了保证各cpp之间能相互访问函数的函数声明。
四个cpp文件：
① cmd_console_tools.cpp：伪图形界面下基本功能函数的具体实现。（不提交）
② hanoi_menu.cpp：菜单的显示与选择。
③ hanoi_multiple_solutions.cpp：菜单中各项汉诺塔演示的实现。
④ hanoi_main.cpp：main函数。
限制：
① 整个程序只允许使用一个递归函数且不超过15行。
② 菜单1~4,6~8输入多个参数必须共用一个函数。
③ 横向输出共用一个函数。
④ 纵向输出共用一个函数。
⑤ 画三根柱子共用一个函数。
⑥ 盘子移动共用一个函数。
⑦ 控制台是Windows控制台主机的旧版控制台，且在选项中去除“快速编辑模式”和“插入模式”。

2. 整体设计思路

如果一上来就让我独自摸索这样一个较为大型的程序，我大概已经在半夜偷偷爬上衷和楼顶准备一跃解千愁了。幸好作业的安排是循序渐进逐层深入的，让我能够像剥洋葱一样一层一层完成这个程序。

首先是main函数。

第一步设置窗口大小。

随后调用menu函数，并在用户输入选项后跳转到对应函数实现相应功能。

其次是menu函数。

输出显示菜单各选项，并传给main函数。需注意每次要清屏。

完成了以上两个准备cpp，接下来就是程序的主要部分，即具体功能的实现。

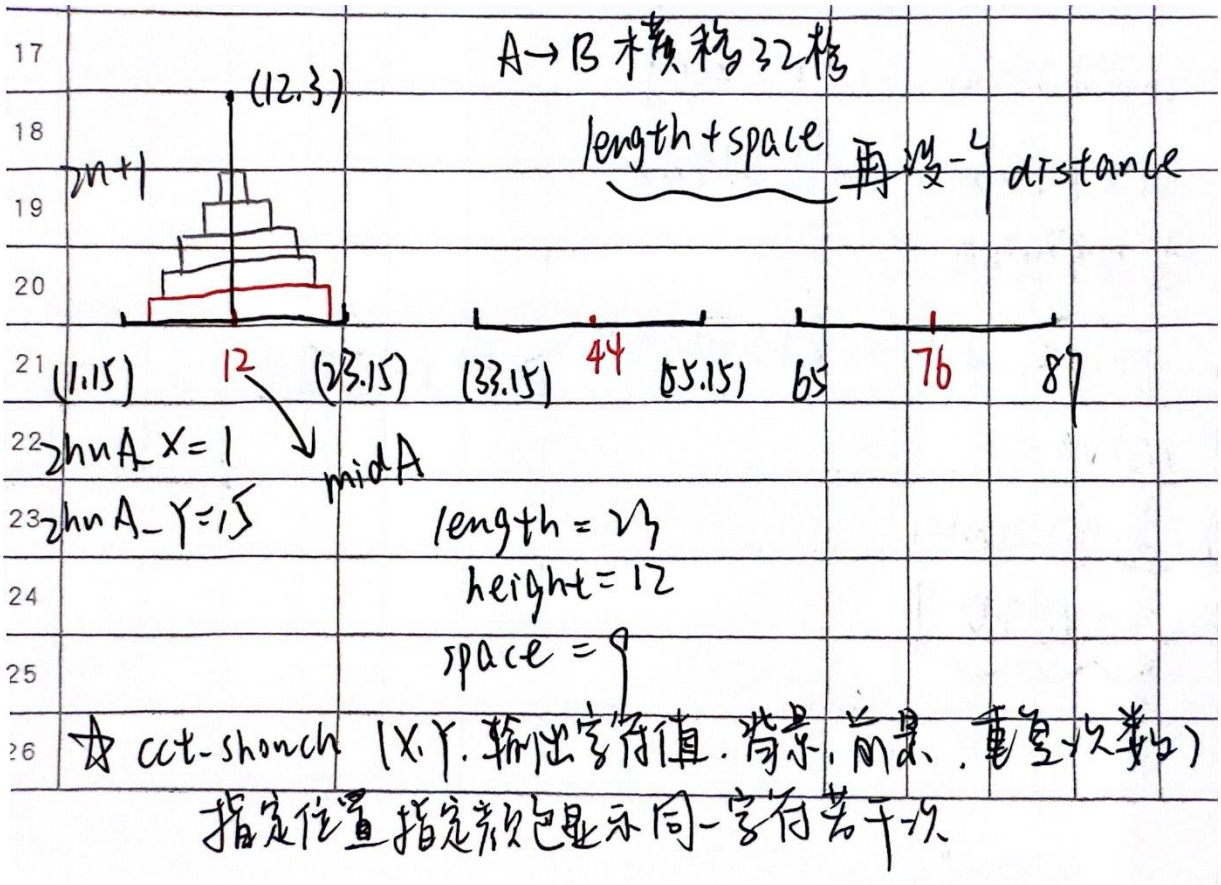
大致思路是：定义两个函数print_heng和print_zong，分别横向打印和纵向打印。再分别写几个函数负责画三个柱子（实现菜单5），在起始柱画初始的n个盘子（实现菜单6），实现盘子移动（菜单7+8），以及主体的初始化函数（进行层数、起始柱、目标柱的输入，错误输入的处理，选择不同菜单时的函数的调用等）。

以及最核心的递归部分，即汉诺塔本身的逻辑。

如果n为1，则直接从起始柱移到目标柱；

如果n>1，则将n-1个盘子从起始柱移动到中间柱，再把第n个盘子移到目标柱，最后把那n-1个盘子移到目标柱。

附上当时读完题后的思维过程：



hanoi_multiple_solutions

只允许一个递归函数 (不超过15行)

菜单 1.2.3.4.8 共用一个递归.

1~4.6~8 输入多个参数共用一个函数

2.4.8 横向输出共用一个

4.8 纵向共用一个

5~9 画三个柱子共用一个

7~9 盘子移动共用一个.

函数定义:

① 横向输出 print_heng

② 纵向输出 print_zong

③ 画3个柱子+画n盘子(n初)

④ 盘子移动.

⑤ print 打印文字移动信息 } 两步一起用过

⑥ move

⑦ 递归 hanoi

⑧ 初始化!

3. 主要功能的实现

3.1.初始化

定义三个全局一维数组 stackA[10], stackB[10], stackC[10]表示圆柱上现有盘子的编号。
定义三个全局简单变量 topA，topB，topC 表示圆柱上现有盘子数量。
initialization 函数：整个程序的初始化，首先对步骤计数和栈顶指针重置，防止多次运行时被上一次的结果影响；然后读取用户输入的层数（n）、起始柱（src）、目标柱（dst），并进行输入错误处理等前序作业已完成的常规操作；再根据用户输入的菜单选项调用相应的函数。

3.2.盘子的移动逻辑+文字解部分

move 函数：更新盘子堆栈和顶部指针，表示盘子从一个柱子移动到另一个柱子。
例：从源柱（假设为 A）移开盘子：stackA[--topA] = 0;
将盘子添加到目标柱（假设为 A）：stackA[topA++] = n;
print 函数：遍历三个一维数组，输出每根柱子上的盘子的编号。
print_heng 函数：进行横向输出打印，根据所选菜单项的不同输出文字与调用函数略有不同，通过 switch-case 语句分情况讨论。
print_zong 函数：进行纵向输出打印，设定列竖式时字母 A 的坐标为基准点，用宏定义表示，并且在有图形解的情况下把基准纵坐标扩大为原来的 2 倍，目的是给图形让出空间。

3.3.图形解部分

3.3.1. 使用工具

利用 cct_gotoxy 函数、cct_showch 函数和 cct_setcolor 函数。
cct_gotoxy 函数：将光标移动到指定位置，输入参数为 X 轴坐标和 Y 轴坐标。
cct_showch 函数：在指定位置，用指定颜色，显示一个字符若干次。输入参数依次为：X 轴坐标、Y 轴坐标、要输出的字符值、背景色、前景色、重复次数。
cct_setcolor 函数：设定指定颜色。输入参数为背景色和前景色。

3.3.2. 具体函数实现

★使用宏定义定义 A 柱基座最左边的点的位置为基准（zhuA_X 和 zhuA_Y），每个基座长 length，两基座间距离 space，每个柱子高度 height，三个基座的中间点横坐标 midA，midB，midC（因为纵坐标都为 zhuA_Y 所以不再另外定义）。
★print_cylinder 函数：用两个循环分别打印出三根柱子+基座。设置延时便于观察出画出的过程。并在最后恢复默认颜色。
★print_plate 函数：在起始柱上画出 n 个盘子。在这里定义一个 num，初始化等于层数 n，用来在循环的时候递减同步修改盘子的颜色。同时定义一个 width 计算盘子的宽度。设置延时便于观察出

画出的过程。并在最后恢复默认颜色。

★plate_move 函数：最初多停留一会，表现出初始化完成后的停顿。定义 top（根据情况指代 topA 或 topB 或 topC）， top_dst（根据情况指代 topA 或 topB 或 topC）， mid（根据情况指代 midA 或 midB 或 midC）， mid_dst;

int* stack = NULL;后续用来指向数组 stackA、B、C。

总体分三个步骤：向上移动，横向平移，向下移动。

共同思路为：清除旧位置盘子图像再绘制新位置盘子图像。

设置延时便于观察出画出的过程。并在最后恢复默认颜色。

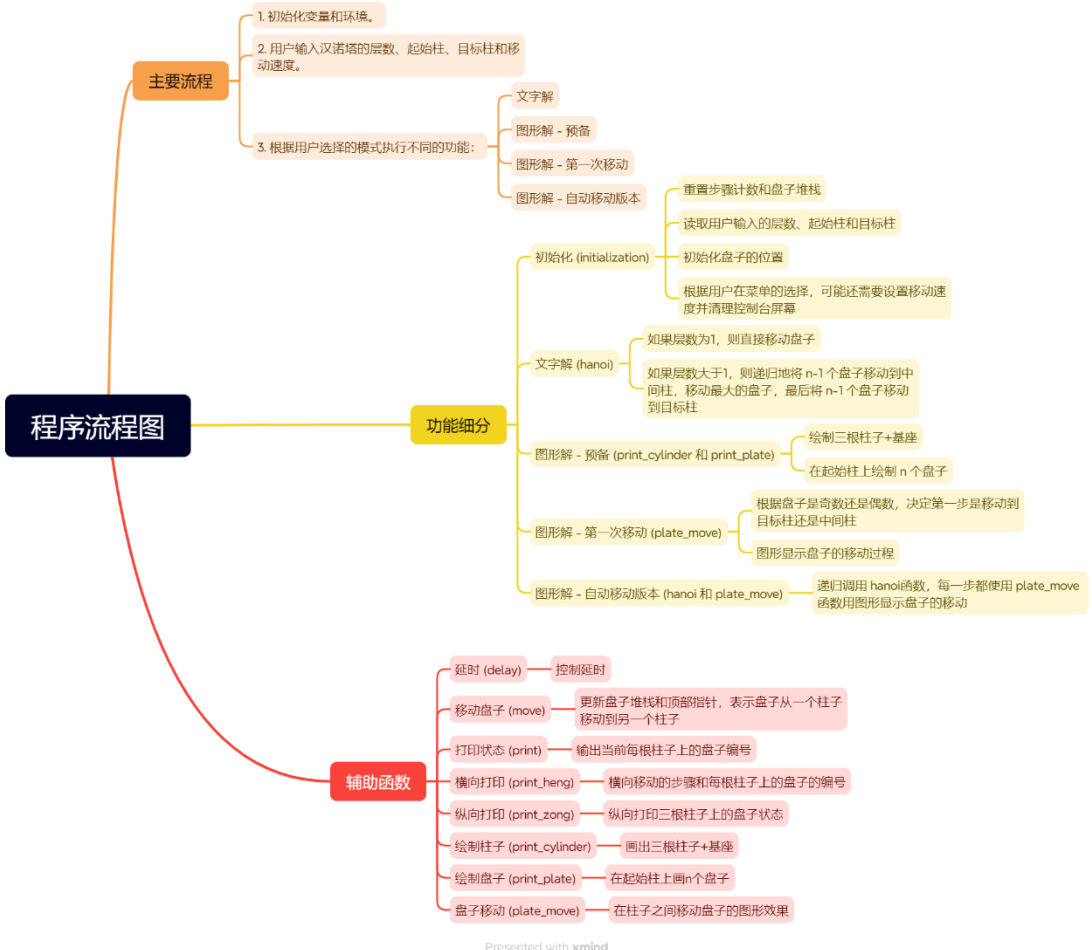
3.4.递归部分

如果n为1，则直接从起始柱移到目标柱；

如果n>1，则将n-1个盘子从起始柱移动到中间柱，再把第n个盘子移到目标柱，最后把那n-1个盘子移到目标柱。

我在print_heng的菜单8情况下加入了调用纵向打印和盘子移动函数，虽然和我最初的命名目的相违背，但是加进来是我能想到的最高效的做法，并且这样做最大的好处是可以减少递归函数的行数（毕竟要求在15行以内）

其他的细节与总体的流程见下图：



4. 调试过程碰到的问题

4.1.使用宏定义的思路的转变

最初的plate_move函数我使用的是穷举法，即src为A时，讨论dst为B或为C；src为B时，讨论dst为A或为C，而且我使用的是确定的数字来表示位置坐标。这样写固然无脑简单，但代码非常冗长，写完之后我自己都看不下去了，最后选了一段头脑比较清醒的时间在草纸上写写画画想清楚后改为使用一个统一的变量或宏定义的形式完成，使其更有普适性也更便于维护。虽然比穷举法要费时费脑一些但我认为是非常有意义且值得的。

4.2.控制递归函数的行数

起初我的print_heng函数的case‘8’中并没有调用print_zong与plate_move函数，因为这与我起这个函数名字的初心相违背。但当我考虑到递归函数时，我发现再特殊处理菜单8的纵向打印与图形部分会让我的代码变长。因此在我考虑之后，我还是选择违背初衷把这两个调用加入到print_heng的case‘8’中，这样我的递归函数中只会出现hanoi和print_heng两个函数，大大减少了代码行数。

4.3.switch-case函数经常忘加break

这个属于我自己的粗心大意带来的问题，但也确实是我经常忽略的一个点。思考之后还是写了进来，希望能让自己长一个教训。

5. 心得体会

5.1.完成本次作业的心得体会

1） 要养成使用宏定义的习惯，特别是对于在程序中多次用到的常量，这样更加便于维护，也方便了后续改写程序适用于其他情况。
2） 要多写注释，方便自己也方便读程序的人。同时写下适当的注释也会让自己思路更加清晰。
3） 开始写之前可以先在草纸上写写画画，构思好大致的思路，充分理解这个程序的目的和要求，并多多思考怎样能尽量做到简洁、高效，减少冗余代码，不要碰到一点思路上的障碍就无脑穷举法。也不要走一步看一步，这样很有可能丢三落四或造成功能的重复与代码的冗余。
4） 变量的命名要清晰、简洁、有意义。
5） 程序比较复杂或庞大时，可以使用多个cpp，并通过头文件对函数进行声明，使得不同cpp文件之间可以相互调用函数，这样写减少了鼠标滚轮反复上下翻动的麻烦，可读性也更高，更清晰明了。
6） 每写一部分就及时调试、测试，以免最后写完发现有问题无处下手。

5.2. 面对复杂程序的体会（若干小题还是一道大题？）

分为若干小题更好。这样我的思路有一个递进的过程，不会感到无从下手。

5.3. 对于重用代码的体会

做最开始的第四章作业并没有考虑那么多，只是抱着一种完成任务的心态，怎么方便怎么来，但幸好那时候的程序还比较简单，没有让我再偷懒的余地。开始做第五章有关汉诺塔的作业的时候就已经开始意识到前面的小题好好完成的重要性，以及前面的小题思考清楚对于后面的帮助。老师对于题目的设置把握的非常完美，既契合当时学习的内容，又有一个逐层递进的逻辑，让我这种脑子笨笨的同学有一个适应的过程。

举例来说，第四章的4-b12和b13让我充分理解了汉诺塔递归部分的逻辑，为我后续编写每步详细过程的横向输出奠定了基础。第五章的5-b6和b7更是为这次大作业埋下了伏笔，让我思考清楚横向打印和纵向打印的逻辑。

对于重用代码，我认为每一次编写程序的时候都要考虑普适性，比如换个类似的场景，我的代码是否能同样适配，或者通过最小的改动就能适配。这样的代码才是好代码。

要想达到这样的目的，首先就是要对代码中多次出现的常量使用宏定义，这样更便于维护。其次是要对有共性的部分进行提取，写成一个独立的函数，这样每次使用只需要一行代码调用这个函数即可。比如我的程序中的move函数和print函数，也是我在5-b6和b7中就在使用的。这两个函数解析了盘子移动时栈顶指针的变化以及每个柱子上盘子编号的打印，非常的实用，后续写代码很多地方直接调用很方便。

5.4. 对于利用函数编写复杂程序的体会

同上题感悟，提取共性的部分写成一个新的函数，这样每次再需要的时候只需要简单一句话调用一下。因此在编写程序的过程中一定要多多思考，思考如何对代码进行提炼，如何用参数满足不同的需求。

6. 附件：源程序

```
    int step = 0;
    int stackA[10] = { 0 }, stackB[10] = { 0 },
    stackC[10] = { 0 };//圆柱上现有圆盘编号
    int topA = 0, topB = 0, topC = 0;//圆柱上现有
    圆盘数量
    static int yanshi = 0;//移动延时
    const int A_X = 10, A_Y = 15;//纵向输出是字母A
    的位置设为基准
    const int zhuA_X = 1, zhuA_Y = 15;//画三根柱子
    时A柱的左下角位置设为基准（A基座最左边的点）
    const int length = 23; // 每个基座长为23
    const int space = 9; // 两个基座之间空9位
    const int height = 12; // 每个柱子高度为12
    const int midA = 12, midB = 44, midC = 76;//三
    个基座的中间点横坐标

/*********************************************
*************************
     函数名称：move
     功    能：进行移动操作

*********************************************
*************************/
void move(int n, char src, char dst)
{
    //从源柱移开盘子
    switch (src)
    {
        case 'A':
            stackA[--topA] = 0;
            break;
        case 'B':
            stackB[--topB] = 0;
```

```
            break;
        case 'C':
            stackC[--topC] = 0;
            break;
        default:
            break;
    }
    //将盘子添加到目标柱
    switch (dst)
    {
        case 'A':
            stackA[topA++] = n;
            break;
        case 'B':
            stackB[topB++] = n;
            break;
        case 'C':
            stackC[topC++] = n;
            break;
        default:
            break;
    }
      }
/*************************************************
****************************
     函数名称：print
     功    能：文字打印每个柱子盘子编号

*************************************************
****************************/
void print()
{
    cout << "A:";
    //遍历stackA数组，打印柱子A上的盘子
    for (int i = 0; i < 10; i++) {
        //如果遇到值为0的元素，就打印两个空格
（表示这个位置没有盘子）
        if (stackA[i] == 0) {
            cout << "  ";
        }
        else {
            cout << setiosflags(ios::right) <<
setw(2) << stackA[i];
        }
    }
    cout << " B:";
    for (int i = 0; i < 10; i++) {
        if (stackB[i] == 0) {
            cout << "  ";
        }
        else {
            cout << setiosflags(ios::right) <<
setw(2) << stackB[i];
        }
    }
    cout << " C:";
    for (int i = 0; i < 10; i++) {
        if (stackC[i] == 0) {
            cout << "  ";
        }
        else {
            cout << setiosflags(ios::right) <<
setw(2) << stackC[i];
        }
    }
    cout << endl;
      }
/*************************************************
****************************
  函数名称：print_heng
  功    能：菜单3、4、8中的横向输出
*************************************************
****************************/
void print_heng(int n, char src, char tmp, char
dst, char choice)
{
    //switch-case别忘了加break啊啊啊！！！！
    switch (choice)
    {
        case '1'://基本解
            cout << setiosflags(ios::right) <<
setw(2) << n << "# " << src << "-->" << dst <<
endl;
            break;
        case '2'://基本解（步数记录）
            step++;
            cout << "第" <<
setiosflags(ios::right) << setw(4) << step << "
步("
                << setw(2) << n << "#: " << src
<< "-->" << dst << ")" << endl;
            break;
        case '3'://内部数组显示（横向）
            step++;
            cout << "第" <<
setiosflags(ios::right) << setw(4) << step << "
步("
                << setw(2) << n << "#: " << src
<< "-->" << dst << ") ";
            move(n, src, dst);
            print();
            break;
        case '4'://内部数组显示（横向+纵向）
            delay();
            cct_gotoxy(0, A_Y + 4);
            step++;
            cout << "第" <<
setiosflags(ios::right) << setw(4) << step << "
```

```
步("
                << setw(2) << n << "#: " << src
<< "-->" << dst << ") ";
            move(n, src, dst);
            print();
            delay();
            print_zong(choice);
            break;
        case '8'://图形解-自动移动版本
            delay();
            cct_gotoxy(0, A_Y * 2 + 4);
            step++;
            cout << "第" <<
setiosflags(ios::right) << setw(4) << step << "
步("
                << setw(2) << n << "#: " << src
<< "-->" << dst << ") ";
            move(n, src, dst);
            print();
            //虽然命名是横向打印，但最后综合考量
还是把菜单8的纵向打印和盘子移动的调用加在了这里
            //加在这里最高效，冗余少，而且能减少
递归函数部分的行数！这是最大的好处
            print_zong(choice);
            move(n, dst, src);
            plate_move(n, src, tmp, dst,
choice);
            move(n, src, dst);
            break;
    }

    }
/*************************************************
*****************************
  函数名称：print_zong
  功    能：菜单4、8中的纵向输出
*************************************************
*****************************/
void print_zong(char choice)
{
    switch (choice)
    {
        case '4'://内部数组显示（横向+纵向）

            //准备好显示三个柱子移动的框架
            cct_gotoxy(A_X - 2, A_Y - 1);
            cout << "======================="
<< endl;
            cct_gotoxy(A_X, A_Y);
            cout << "A         B         C";
            //在对应位置打印盘子号
            for (int i = 0; i < 10; i++) {
                cct_gotoxy(A_X - 1, A_Y - 2 -
i);
                //如果遇到值为0的元素，就打印两
个空格（表示这个位置没有盘子）
                if (stackA[i] == 0) {
                    cout << "  ";
                }
                else {
                    cout <<
setiosflags(ios::right) << setw(2) << stackA[i];
                }
            }
            for (int i = 0; i < 10; i++) {
                cct_gotoxy(A_X - 1 + 10, A_Y - 2
- i);
                if (stackB[i] == 0) {
                    cout << "  ";
                }
                else {
                    cout <<
setiosflags(ios::right) << setw(2) << stackB[i];
                }
            }
            for (int i = 0; i < 10; i++) {
                cct_gotoxy(A_X - 1 + 20, A_Y - 2
- i);
                if (stackC[i] == 0) {
                    cout << "  ";
                }
                else {
                    cout <<
setiosflags(ios::right) << setw(2) << stackC[i];
                }
            }
            break;
        case '8':
            //基准纵坐标变为原来的2倍，给柱子图
形部分让出位置
            //准备好显示三个柱子移动的框架
            cct_gotoxy(A_X - 2, A_Y * 2 - 1);
            cout << "======================="
<< endl;
            cct_gotoxy(A_X, A_Y * 2);
            cout << "A         B         C";
            //在对应位置打印盘子号
            for (int i = 0; i < 10; i++) {
                cct_gotoxy(A_X - 1, A_Y * 2 - 2
- i);
                //如果遇到值为0的元素，就打印两
个空格（表示这个位置没有盘子）
                if (stackA[i] == 0) {
                    cout << "  ";
                }
                else {
                    cout <<
setiosflags(ios::right) << setw(2) << stackA[i];
```

```
            }
        }
        for (int i = 0; i < 10; i++) {
            cct_gotoxy(A_X - 1 + 10, A_Y * 2 - 2 - i);
            if (stackB[i] == 0) {
                cout << "  ";
            }
            else {
                cout << setiosflags(ios::right) << setw(2) << stackB[i];
            }
        }
        for (int i = 0; i < 10; i++) {
            cct_gotoxy(A_X - 1 + 20, A_Y * 2 - 2 - i);
            if (stackC[i] == 0) {
                cout << "  ";
            }
            else {
                cout << setiosflags(ios::right) << setw(2) << stackC[i];
            }
        }
        break;
    }//switch
    }
/****************************************************************************
  函数名称：print_cylinder
  功    能：在屏幕上画出三根柱子（菜单5）
****************************************************************************/
void print_cylinder()
{
    //柱子的基座部分
    for (int i = 0; i < 3; i++)
    {
        cct_showch(zhuA_X + i * (length + space), zhuA_Y, ' ', 14, 14, length);
    }
    //柱子的柱部分
    for (int j = 0; j < height; j++)
    {
        for (int i = 0; i < 3; i++) {
            cct_showch(zhuA_X + i * (length + space) + length / 2, zhuA_Y - j - 1, ' ', 14, 14, 1);
            Sleep(50);//设置一个小延时
        }
    }
    cct_gotoxy(0, 40);
    cct_setcolor(0, 7);//恢复默认颜色
}

/****************************************************************************
  函数名称：print_plate
  功    能：在起始柱上画n个盘子（初始状态）（菜单6）
****************************************************************************/
void print_plate(char src, int n)
{
    int num = n;//用来递减同步修改盘子颜色
    int start_X = zhuA_X + (src - 'A') * (length + space); // 当前柱子的起始X坐标
    // 在指定柱子上绘制n个盘子，从底部开始
    for (int i = 0; i < n; i++) {
        int width = n * 2 + 1 - i * 2; // 计算盘子的宽度
        cct_showch(start_X + (length / 2 - width / 2), zhuA_Y - 1 - i, ' ', num, num, width);
        num--;
        Sleep(50);//设置一个小延时
    }
    cct_gotoxy(0, 40);
    cct_setcolor(0, 7);//恢复默认颜色
    }

/****************************************************************************
  函数名称：plate_move
  功    能：在柱子间移动盘子（菜单7~9）
****************************************************************************/
void plate_move(int n, char src, char tmp, char dst, char choice)
{
    Sleep(1000);//初始化完成后多停留一会
    int top = 0, top_dst = 0, mid = 0, mid_dst = 0;
    int* stack = NULL;//后续用来指向数组stackA、B、C
    switch (src)
    {
        case 'A':
            top = topA;
            mid = midA;
            stack = stackA;
            break;
        case 'B':
            top = topB;
            mid = midB;
            stack = stackB;
            break;
        case 'C':
            top = topC;
            mid = midC;
```

```
            stack = stackC;
            break;
    }
    switch(dst)
    {
        case 'A':
            mid_dst = midA;
            top_dst = topA;
            break;
        case 'B':
            mid_dst = midB;
            top_dst = topB;
            break;
        case 'C':
            mid_dst = midC;
            top_dst = topC;
            break;
    }

    //向上移动
    for (int i = 0; i + top < height; i++)
    {
        cct_showch(mid, zhuA_Y - top - i, ' ', 0, 0, 1);// 清除柱子顶部标记
        //清除旧位置
        cct_showch(mid - stack[top - 1], zhuA_Y - top - i, ' ', 0, 0, stack[top - 1] * 2 + 1);
        //绘制新位置
        cct_showch(mid, zhuA_Y - top - i, ' ', 14, 14, 1);// 绘制柱子顶部标记
        cct_showch(mid - stack[top - 1], zhuA_Y - top - i - 1, ' ', stack[top - 1], stack[top - 1], stack[top - 1] * 2 + 1);
        cct_showch(mid - length / 2 + 1, 3, ' ', 0, 0, 21);//消掉最上面剩余的
        Sleep(50);
    }

    //横向移动
    int distance = abs(dst - src) * (length + space);//横向移动距离
    for (int i = 0; i < distance; i++)
    {
        //清除旧位置的盘子图像
        cct_showch(mid - stack[top - 1] + (dst - src > 0 ? 1 : -1) * i, 1, ' ', 0, 0, stack[top - 1] * 2 + 1);
        //绘制新位置的盘子图像
        if (i < distance)  //在最后一次迭代时不再绘制新位置
        {
            cct_showch(mid - stack[top - 1] +(dst-src>0?1:-1)*(i+1), 1, ' ', stack[top - 1], stack[top - 1], stack[top - 1] * 2 + 1);
        }
        Sleep(50);
    }
    cct_showch(mid + (dst - src > 0 ? 1 : -1) * distance - stack[top - 1], 1, ' ', 0, 0, stack[top - 1] * 2 + 1);

    //向下移动
    for (int i = 0; i < height - 1 - top_dst; i++)
    {
        cct_showch(mid_dst, 3, ' ', 14, 14, 1);
        //清除旧位置
        cct_showch(mid_dst - stack[top - 1], 1 + i + 2, ' ', 0, 0, stack[top - 1] * 2 + 1);
        //如果不是第一步，也清除柱子顶部的标记
        if (i != 0)
        {
            cct_showch(mid_dst, 1 + i + 2, ' ', 14, 14, 1);
        }
        //绘制新位置
        cct_showch(mid_dst - stack[top - 1], 1 + i + 3, ' ', stack[top - 1], stack[top - 1], stack[top - 1] * 2 + 1);
        Sleep(50);
    }

    cct_gotoxy(0, 35);
    cct_setcolor(0, 7); // 恢复默认颜色
}

/****************************************************************************
  函数名称：hanoi
  功    能：递归函数
****************************************************************************/
void hanoi(int n, char src, char tmp, char dst, char choice)
{
    if (n == 1)
        print_heng(n, src, tmp, dst, choice);
    else
    {
        hanoi(n - 1, src, dst, tmp, choice);
        print_heng(n, src, tmp, dst, choice);
        hanoi(n - 1, tmp, src, dst, choice);
    }
}
```