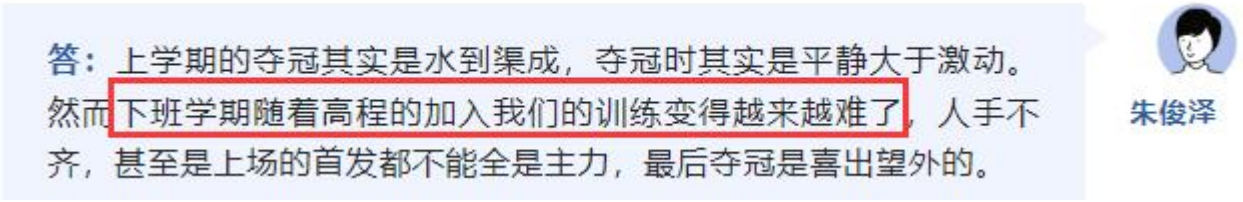
同济大学 实验报告

高程汉诺塔大作业



（高程磨练了我们球队的意志力，帮助我们济勤连夺两连冠！谢谢老师）

姓名：朱俊泽

学号：2351114

班级：信15

1. 题目

1.1.要求

1.1.1. 题目要求简述

题目第一项要求就是打印出汉诺塔的基本步骤。
题目第二项就是第一项加上步数的记录。
题目第三项就是显示汉诺塔具体步骤的数组展示
题目第四项就是整个汉诺塔项目的简单可视化---数组纵向和横向显示
题目第五、六、七项就是初步的应用cct包来打印三个基础柱，已经第一次盘子的堆放，第一个移动
题目第八项就是汉诺塔最简单解法的前七项题目的总和
题目第九项要求实现一个能够自行操作的汉诺塔游戏，在汉诺塔游戏的基础上增加了输入环节。

2.整体设计思路

2.1 菜单函数部分

使用循环输入，用getch来避免输出显示，然后返回这个值（char）

2.2 主函数部分

使用一个op变量来接受menu菜单函数返回输入的值，分情况进入一个叫做game的函数，在这个game函数结束的之后用一个循环控制只有输入回车的时候才能进行下一个循环。

2.3 头函数部分

将所有multiple_solutions.cpp种命名的函数和menu菜单函数先在hanoi.h头文件中申明，避免调用问题。

2.4 主要实现函数部分（multiple_solutions.cpp）

2.4.1 内部主要逻辑

首先通过主函数switch进入一个game函数部分，分情况处理输入，比如在4和8选项有速度的输入之类的。然后从game函数中去执行hanoi函数，hanoi就是主要的递归函数，其中包括hanoi往下递归，也包括主要的运作函数going，特别的。我为第九项准备了一个play函数来运行。

2.4.2 申明的函数意义

void hanoi(int n, char src, char tmp, char dst, int op);这是主要的递归函数
void game(int op);这个是 main 函数进入的主要逻辑来分配情况，控制输入问题
void pause();这个用于暂停程序
void going(int n, char src, char tmp, char dst, int op);这个用于处理汉诺塔函数中需要运行的程序具体内容
void move(int n, char src, char dst);这个用于处理移动之后的三个一维数组和 top 值变化
void print_disk();这个用于打印三个基础柱子
void play(int n, char src, char tmp, char dst, int op, int end, char end_char);这个用于专项处理第九项题目的问题
bool checkend(int n, char dst);这个用于判断第九项中游戏是否已经成功结束
void disk_move(int n, char src, char dst, int op);这个用于演示盘子移动的全过程
void print_disk_ini(char src, int sum);这个用于生成最开始的盘子
void print_tower();这个用于打印数组纵向展示
void print_array_ini();这个用于打印数组初始状态横向展示
void print_array();这个用于打印数组横向展示
void print_move(int n, char src, char tmp, char dst);这个用于打印移动的基础过程，对应 1，2,3,4，项内容

void print_cnt(int n);这个用于打印记录好的步数

3.主要功能的实现

3.1 菜单函数部分

正常清屏然后用循环提示输入0-9即可，控制输入正确范围再返回出输入值。

3.2 主函数部分

接受了菜单函数的返回值之后分情况进入game函数内设置好的情况

3.3 主要实现函数部分（multiple_solutions.cpp）

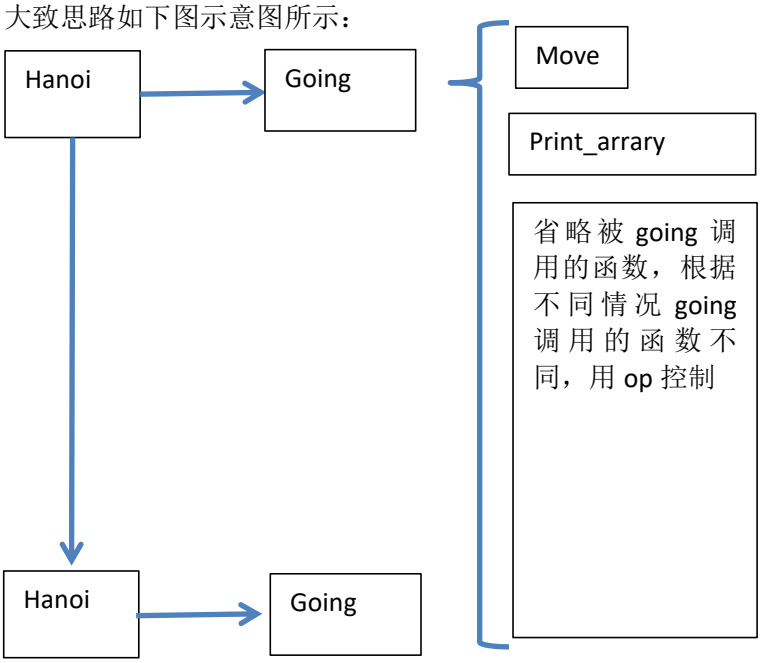
3.3.1 主要函数的核心逻辑部分--对应函数：

Hanoi，going

这两个函数由hanoi决定递归方向，going决定每次递归具体执行的逻辑调用的函数。

把going函数嵌套在hanoi之中就可以提高代码的复用性，让going函数决定调用哪一些提前准备好的函数组，修改函数也会简单一点。

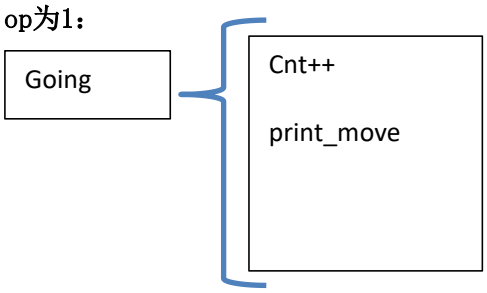
大致思路如下图示意图所示：



省略递归

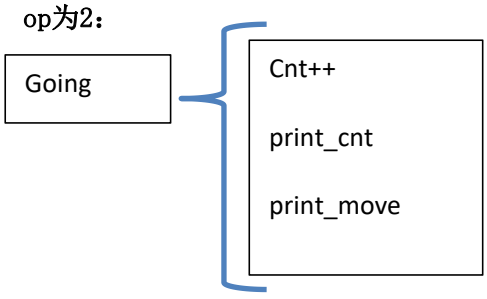
以下是不同op时going对应调用的函数，对应函数解释参考下文

op为1：



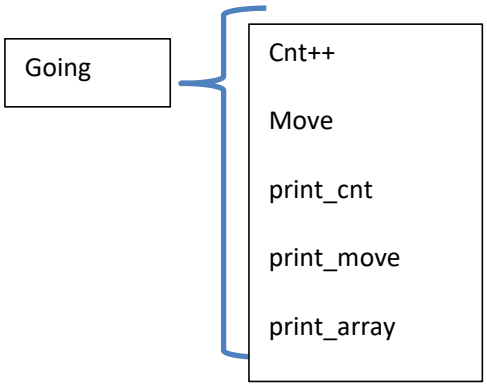
只需要调用print_move函数来展示汉诺塔的基本移动就可以

op为2：



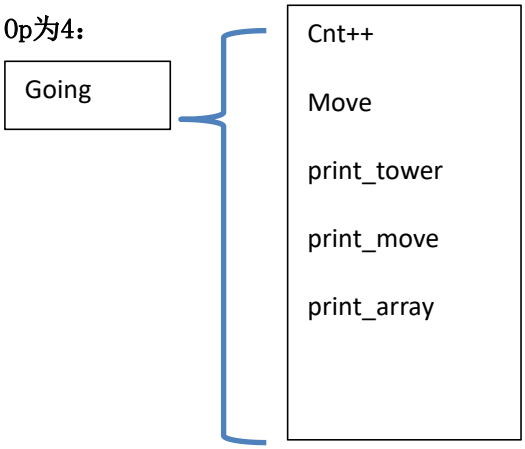
调用print_cnt先展示步数再print_move

op为3：



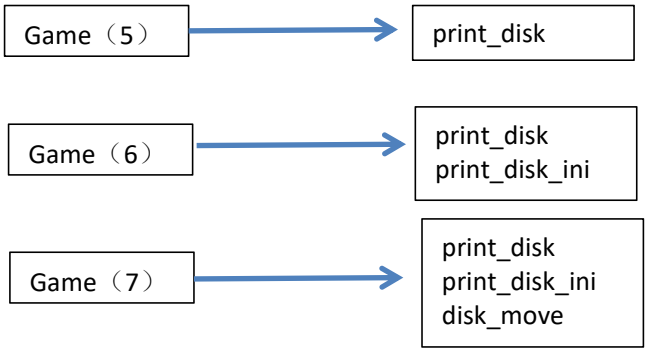
调用move先处理数组的移动，再print_cnt展示步数，然后print_move打印基本移动，再print_array数组横向展示。

0p为4：



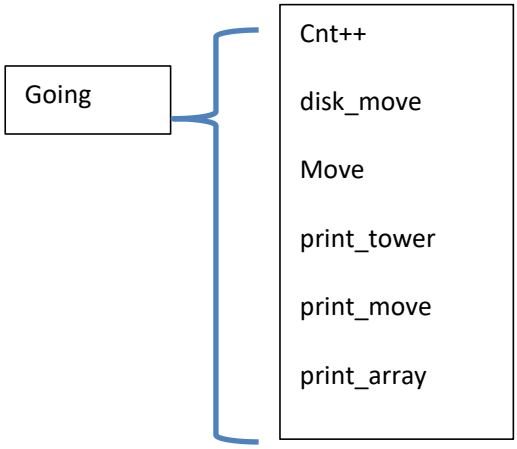
调用move先处理数组的移动，再print_tower数组纵线展示，再进行print_array的横向展示。

0p为5, 6, 7：



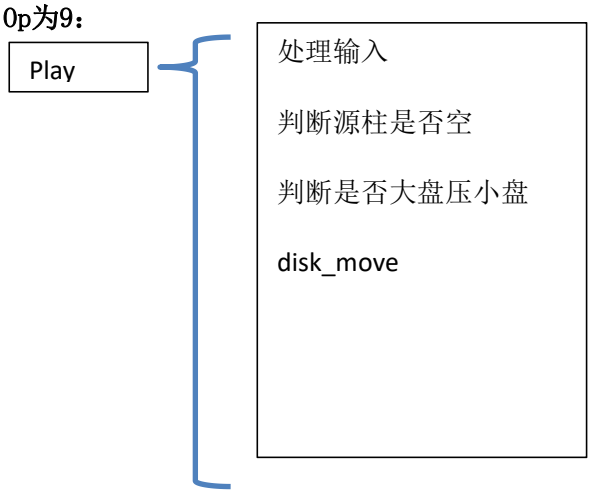
此时不需要处理，在game中已经处理。

Op为8：



一定要先进行disk_move对盘进行一次移动，然后move函数处理对应数组，否则会出现下面调试当中的问题1.更新了可视化的盘子之后，再进行print_tower和print_array进行横向展示纵向展示

Op为9：



此时不需要处理，有一个专门的play函数进行处理。

3.3.2 数组表示的hanoi的基础移动--对应函数：

move,print_array,print_move,print_cnt,print_array_ini,print_tower

命名好全局的一维数组A[],B[],C[]用来表示三个柱子上的具体状态,全局向量topa,topb,topc用来表示三个柱子的顶端问题，还有全局向量cnt用来记录步数。

Move：

用move函数来处理每一步移动，move传入的，n，src，dst分别代表移动的盘子序号，源头，目标柱。分情况处理三个一维数组的顶端和top值。

print_array：

用三个for循环，如果数组的某个值为0，输出两个空格，可以覆盖上一次的内容，如果不是，输出数组的

对应值。

print_move：

输入n代表盘子序号，src源头，dst目标柱。则输出一个标准的汉诺塔移动

print_cnt：

输出全局变量cnt代表移动的步数。

print_array_ini：

输出初始化的横向数组，和print_array的主要形式差不多。

print_tower：

用三个for循环配合上cct_gotoxy的函数输出了数组的纵向展示。

3.3.3 hanoi的移动可视化--对应函数：

print_disk，print_disk_ini，disk_move，

Print_disk：

利用两个for循环函数，结合cct_showch函数，采用14号色块--亮黄色色块，其实大于10应该就行，因为之后汉诺塔盘子初始化中最多10个盘子，颜色的选择下文会提到

print_disk_ini：

打印盘子初始化，颜色的选择这个方面很有说法，比如我想初始化在a柱上的n个柱子（n<=10）的话，盘子的宽度其实就是2*盘子序号+1，至于颜色的挑选，我直接让盘子的颜色等于盘子的序号，之后也按照这个规定去规范盘子的颜色。cct_showch的参数第一个是开始打印的坐标，只需要是柱子最中心的坐标减去盘子序号就可以了。这样打印就会刚好是一个等差数列，并且关于柱子对称。

Disk_move：

最重量级的来了，就是这个disk_move函数，我没有什么好的思路，直接对src和dst做了枚举3*3中情况，因为不会出现自己往自己移动所以最后六类情况我都枚举了，首先先判断传入参数dst，分三个情况也就是a，b，c这里不一一赘述了，当判断好了源头柱子后，先进行一个操作把盘子从当前柱子上移动到最上空，for循环控制执行的次数，就是往上移动的次数，是15-top次，每次循环先cct_showch把盘子所处位置涂黑色，再给柱子涂一个亮黄色，再给i-1高度涂成盘子的颜色，具体颜色根据数组X【topx】决定，因为这个就是盘子的序号，也就是遵守了之前设置下来的规定，用盘子的序号来确定盘子颜色，宽度也就是2*X【topx】+1，在上文也表达过。这样以来盘子就上升到了源头柱子的最上空。之后根据源头柱和目标柱之间横坐标关系先涂黑色然后移动涂色，这样以来盘子就移动到了目标柱子的最上空，然后根据目标柱上面X【topx】决定要下降的操作要执行多少次，用for循环控制，注意移动后要重新打印亮黄色的柱子！

3.3.4 项目⑨的对应play函数：

首先输入问题很值得考虑，先输入src，再输入dst，这时进入一个循环，表示剩下多输入的，如果是回车就结束，这样就控制好了只能输入两次，如果有问题，continue重新开始循环，每次循环开始会在输入的位置和第二行全部涂黑色，也就是清除了输入缓存。当输入没有问题后，就可以判断第一个原柱子是否为空的问题，这个问题判断第一个源头柱子的topx即可，然后到了大盘压小盘的问题，只要判断src【topsrc】和dst【topdst】即可。如果输入正常，此时可以进行调用move，disk_move，还有print_array和print_tower，还有一个关键的checkend函数，如果dst目标柱子的topx等于一开始输入的n，就可以判断游戏结束了。

4. 调试过程碰到的问题

4.1 问题一

4.1.1 问题简述：

其实就有一个非常非常大的问题，就是在编写项目9的程序时，play函数中我需要进行disk_move(n, src, dst, op)，这里的n其实是代表盘子的序号，在正常的hanoi的递归中，每一次都是最优解，也就是每一次的n其实都是固定的，我如果像最优解的hanoi一样编写play函数，那么如果玩家输入一次ac，再输入ca也就是做了一次无用的操作，那就会打乱后面所有的hanoi的盘子序号，颜色和宽度和位置都会出现问题。

4.1.1 问题解决：

这时候需要我把n代换成对应的src【topsrc】，类似下图，就会解决这个问题。

```
if (src == 'A' && dst == 'B')
{
    n = A[topa];
```

5. 心得体会

5.1 心得体会，感受

写如此大的一个项目，需要对代码分块为函数进行谨慎考虑，如果分块工作做得不好，会使得程序的修改和debug非常的困难。我想这就是老师说的耦合性吧。

5.2 若干小题还是一道大题好？

我觉得还是若干小题比较好，这样直接帮助我们考虑函数的复用性，第一和二项要求我写的函数一路用到底了，帮助我们提高函数的复用性，提高程序耦合性。

5.3 如何更好复用代码？

我觉得就是要把作业拆开成不同的小问题，如果从一整个大问题来看的话就会不由自主的线性的编写这个程序。对每一个函数进行功能的设定，对小问题的要求把线性的程序分块成不同的函数。

5.4 如何更好利用函数？

我觉得就是要把问题分解为不同小问题，观察这些问题的共同之处，把这些共同之处分块为函数，剩下的可以考虑直接线性的编程或者也写成函数。总之我觉得就是：要自上而下的考虑问题，先考虑好我能不能把问题分解成小问题，小问题之间有没有共同点，进行自上而下的编程。可以先把函数的框架写出来，也就是先把函数名，函数作用表列出来。

6. 附件：源程序

Hanoi

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

Going

```
void going(int n, char src, char tmp, char dst, int op)
{
    switch (op)
    {
        case 1:
            cnt++;
            print_move(n, src, tmp, dst);
            cout << endl;
            break;
        case 2:
            cnt++;
            print_cnt(n);
            cout << "( ";
            print_move(n, src, tmp, dst);
            cout << ")" << endl;
            break;
        case 3:
            cnt++;
            move(n, src, dst);
            print_cnt(n);
            cout << "( ";
            print_move(n, src, tmp, dst);
            cout << ")  ";
            print_array();
            cout << endl;
            break;
        case 4:
            cnt++;
            move(n, src, dst);
            print_tower();
            cct_gotoxy(0, 17+20);
            cout << "第" << setw(4) << cnt << "步" << "(" << n << "#: " << src << "-->" << dst << ")  ";
            print_array();
            pause();
            break;
        case 8:
            cnt++;
            disk_move(n, src, dst, op);
            move(n, src, dst);
            print_tower();
            cct_gotoxy(0, 17 + 20);
            cout << "第" << setw(4) << cnt << "步" << "(" << n << "#: " << src << "-->" << dst << ")  ";
            print_array();

            pause();
            break;
        default:
            break;
    }
}
```

Move

print_array

```
void move(int n, char src, char dst)
{
    switch (src)
    {
        case 'A':
            A[topa--] = 0;
            break;
        case 'B':
            B[topb--] = 0;
            break;
        case 'C':
            C[topc--] = 0;
            break;
        default:
            break;
    }
    switch (dst)
    {
        case 'A':
            A[++topa] = n;
            break;
        case 'B':
            B[++topb] = n;
            break;
        case 'C':
            C[++topc] = n;
            break;
        default:
            break;
    }
}
```

```
void print_array()
{
    cout << "A:";
    for (int i = 1; i <= 10; i++)
    {
        if (A[i] == 0)
            cout << "  ";
        else
            cout << setw(2) << A[i];
    }
    cout << "B:";
    for (int i = 1; i <= 10; i++)
    {
        if (B[i] == 0)
            cout << "  ";
        else
            cout << setw(2) << B[i];
    }
    cout << "C:";
    for (int i = 1; i <= 10; i++)
    {
        if (C[i] == 0)
            cout << "  ";
        else
            cout << setw(2) << C[i];
    }
}
```

print_move

```
void print_move(int n, char src, char tmp, char dst)
{
    cout << n << "# " << src << "---->" << dst;
}
```

print_cnt

```
void print_cnt(int n)
{
    cout << "第" << setw(4) << cnt << " 步";
}
```

print_array_ini

```
void print_cnt(int n)
{
    cout << "第" << setw(4) << cnt << " 步";
}
```

print_tower

```
void print_tower()
{
    cct_gotoxy(0, 11+20);
    cout << "========================" << endl;
    cct_gotoxy(0, 12+20);
    cout << "A          B          C";
    for (int i = 1; i <= 10; i++)
    {
        cct_gotoxy(0, 20+11 - i);
        if (A[i] != 0)
        {
            cout << setw(2) << A[i];
        }
        else
        {
            cout << setw(2) << " ";
            break;
        }
    }
    for (int i = 1; i <= 10; i++)
    {
        cct_gotoxy(10, 11+20 - i);
        if (B[i] != 0)
        {
            cout << setw(2) << B[i];
        }
        else
        {
            cout << setw(2) << " ";
            break;
        }
    }
```

print_disk

```
for (int i = 1; i <= 10; i++)
{
    cct_gotoxy(20, 11+20 - i);
    if (C[i] != 0)
    {
        cout << setw(2) << C[i];
    }
    else
    {
        cout << setw(2) << " ";
        break;
    }
}
```

print_disk_ini

```
void print_disk()
{
    for (int i = 0; i < 3; i++)
    {
        cct_showch(2 + 30 * i, 17, 0, 14, 14, 25);
    }
    for (int j = 0; j < 15; j++)
    {
        cct_showch(14, 17 - j, 0, 14, 14, 1);
        cct_showch(44, 17 - j, 0, 14, 14, 1);
        cct_showch(74, 17 - j, 0, 14, 14, 1);
        Sleep(50);
    }
    cct_gotoxy(0, 35);
    cct_setcolor(0, 7);
}
```

```
void print_disk_ini(char src, int sum)
{
    int sum1 = sum;
    if (src == 'A')
    {
        for (int i = 0; i < sum; i++)
        {
            cct_showch(14 - sum1, 16 - i, 0, sum1, sum1, sum1 * 2 + 1);
            Sleep(50);
            sum1--;
        }
    }
    else if (src == 'B')
    {
        for (int i = 0; i < sum; i++)
        {
            cct_showch(44 - sum1, 16 - i, 0, sum1, sum1, sum1 * 2 + 1);
            Sleep(50);
            sum1--;
        }
    }
    else if (src == 'C')
    {
        for (int i = 0; i < sum; i++)
        {
            cct_showch(74 - sum1, 16 - i,  0, sum1, sum1, sum1 * 2 + 1);
            Sleep(50);
            sum1--;
        }
    }
    cct_gotoxy(0, 35);
    cct_setcolor(0, 7);
}
```

disk_move

```
void disk_move(int n, char src, char dst, int op)
{
    cct_setcursor(CURSOR_INVISIBLE);
    if (src == 'A')
    {
        for (int i = 17 - topa; i > 2; i--)
        {
            cct_showch(14- A[topa], i, 0, 0, 0, A[topa] * 2 + 1);
            cct_showch(14, i, 0, 14, 14,  1);
            cct_showch(14 - A[topa], i-1, 0, A[topa], A[topa], A[topa] * 2 + 1);
            Sleep((6 - speed) * 30);
        }
        if (dst == 'C')
        {
            for (int i = 0; i < 60; i++)
            {
                cct_showch(14 - A[topa] + i, 2, 0, 0, 0, A[topa] * 2 + 1);
                cct_showch(14 - A[topa] + 1 + i, 2, 0, A[topa], A[topa], A[topa ] * 2 + 1);
                Sleep((6 - speed) * 30);
            }
            for (int i = 0; i < 14 - topc; i++)
            {
                cct_showch(74 - A[topa ], 2 + i, 0, 0,0, A[topa] * 2 + 1);
                cct_showch(74 - A[topa ], 2 + i + 1, 0, A[topa], A[topa ], A[topa] * 2 + 1);
                if (i != 0)
                    cct_showch(74, 2 + i, 0, 14, 14, 1);
                Sleep((6 - speed) * 30);
            }
        }
        else if (dst == 'B')
        {
            for (int i = 0; i < 30; i++)
            {
                cct_showch(14 - A[topa] + i, 2, 0, 0, 0, A[topa] * 2 + 1);
                cct_showch(14 - A[topa] + 1 + i, 2, 0, A[topa], A[topa], A[topa] * 2 + 1);
                Sleep((6 - speed) * 30);
            }
            for (int i = 0; i < 14 - topb; i++)
            {
                cct_showch(44 - A[topa], 2 + i, 0, 0, 0, A[topa] * 2 + 1);
                cct_showch(44 - A[topa], 2 + i + 1, 0, A[topa], A[topa], A[topa] * 2 + 1);
                if (i != 0)
                    cct_showch(44, 2 + i, 0, 14, 14, 1);
                Sleep((6 - speed) * 30);
            }
        }
    }
    if (src == 'B')
    {
        for (int i = 17 - topb; i > 2; i--)
        {
            cct_showch(44 - B[topb], i, 0, 0, 0,B[topb] * 2 + 1);
            cct_showch(44, i, 0, 14, 14, 1);
            cct_showch(44 - B[topb], i - 1, 0, B[topb], B[topb], B[topb] * 2 + 1);
            Sleep((6 - speed) * 30);
        }
        if (dst == 'C')
        {
            for (int i = 0; i < 30; i++)
            {
                cct_showch(44 - B[topb ] + i, 2, 0, 0, 0, B[topb ] * 2 + 1);
                cct_showch(44 - B[topb] + 1 + i, 2, 0, B[topb ], B[topb], B[topb ] * 2 + 1);
                Sleep((6 - speed) * 30);
            }
            for (int i = 0; i < 14 - topc; i++)
            {
                cct_showch(74 - B[topb ], 2 + i, 0, 0, 0, B[topb ] * 2 + 1);
                cct_showch(74 - B[topb ], 2 + i + 1, 0, B[topb ], B[topb ], B[topb ] * 2 + 1);
                if (i != 0)
                    cct_showch(74, 2 + i, 0, 14, 14, 1);
                Sleep((6 - speed) * 30);
            }
        }
        else if (dst == 'A')
        {
            for (int i = 0; i < 30; i++)
            {
                cct_showch(44 - B[topb] - i, 2, 0, 0, 0, B[topb] * 2 + 1);
                cct_showch(44 - B[topb] - 1 - i, 2, 0, B[topb], B[topb], B[topb] * 2 + 1);
                Sleep((6 - speed) * 30);
            }
            for (int i = 0; i < 14 - topa; i++)
            {
                cct_showch(14 - B[topb], 2 + i, 0, 0, 0, B[topb] * 2 + 1);
                cct_showch(14 - B[topb], 2 + i + 1, 0, B[topb], B[topb], B[topb] * 2 + 1);
                if (i != 0)
                    cct_showch(14, 2 + i, 0, 14, 14, 1);
                Sleep((6 - speed) * 30);
            }
        }
    }
    if (src == 'C')
    {

        for (int i = 0; i + topc < 15; i++)
        {
            cct_showch(74 - C[topc], 17 - topc - i, 0, 0, 0, C[topc] * 2 + 1);
            cct_showch(74, 17 - topc - i, 0, 14, 14, 1);
            cct_showch(74 - C[topc], 16 - topc - i, 0, C[topc], C[topc], C[topc] * 2 + 1);
            Sleep((6 - speed) * 30);
        }
        if (dst == 'B')
        {
            for (int i = 0; i < 30; i++)
            {
                cct_showch(74 - C[topc] - i, 2, 0, 0, 0, C[topc] * 2 + 1);
                cct_showch(74 - C[topc] - 1 - i, 2, 0, C[topc], C[topc], C[topc] * 2 + 1);
                Sleep((6 - speed) * 30);
            }
            for (int i = 0; i < 14 - topb; i++)
            {
                cct_showch(44 - C[topc], 2 + i, 0, 0, 0, C[topc] * 2 + 1);
                cct_showch(44 - C[topc], 2 + i + 1, 0, C[topc], C[topc], C[topc] * 2 + 1);
                if (i != 0)
                    cct_showch(44, 2 + i, 0, 14, 14, 1);
                Sleep((6 - speed) * 30);
            }
        }
        else if (dst == 'A')
        {
            for (int i = 0; i < 60; i++)
            {
                cct_showch(74 - C[topc] - i, 2, 0, 0, 0, C[topc] * 2 + 1);
                cct_showch(74 - C[topc] - 1 - i, 2, 0, C[topc], C[topc], C[topc] * 2 + 1);
                Sleep((6 - speed) * 30);
            }
            for (int i = 0; i < 14 - topa; i++)
            {
                cct_showch(14 - C[topc], 2 + i, 0, 0, 0, C[topc] * 2 + 1);
                cct_showch(14 - C[topc], 2 + i + 1, 0, C[topc], C[topc], C[topc] * 2 + 1);
                if (i != 0)
                    cct_showch(14, 2 + i, 0, 14, 14, 1);
                Sleep((6 - speed) * 30);
            }
        }
    }
    cct_gotoxy(0, 35);
    cct_setcolor(0, 7);
}
```

Play

```
void play(int n, char src, char tmp, char dst, int op,int end,char end_char)
{
    int x, y;
    cct_getxy(x, y);
    while (1)
    {
        cct_setcursor(6);
        cct_showch(x, y, 32, 0, 0, 10);
        cct_showch(0, y + 1, 32, 0, 0, 25);
        cct_setcolor(0, 7);
        cct_gotoxy(x, y);
        cin.get(src);
        if (src == 'Q' || src == 'q')
        {
            cout << "游戏中止!!!!!" << endl;
            Sleep(500);
            return;
        }
        cin.get(dst);
        int cishu = 0;
        for (; getchar() != '\n'; cishu++);
        if (cishu > 0)
            continue;
        if (src == 'A' || src == 'B' || src == 'C' || src == 'a' || src == 'b' || src == 'c')
        {
            if (src >= 'a' && src <= 'c')
                src -= 32;
            if (src == 'A' && dst != 'A')
            {
                if (topa == 0)
                {
                    cout << "原柱为空！" << endl;
                    Sleep(900);
                    continue;
                }
            }
            if (src == 'B' && dst != 'B')
            {
                if (topb == 0)
                {
                    cout << "原柱为空！" << endl;
                    Sleep(900);
                    continue;
                }
            }
            if (src == 'C' && dst != 'C')
            {
                if (topc == 0)
                {
                    cout << "原柱为空！" << endl;
                    Sleep(900);
                    continue;
                }
            }
```

```
            continue;
        if (dst == 'A' || dst == 'B' || dst == 'C' || dst == 'a' || dst == 'b' || dst == 'c')
        {
            if (dst >= 'a' && dst <= 'c')
                dst -= 32;
            if (dst == src)
                continue;
        }
        else
            continue;
        if (src == 'A' && dst == 'B')
        {
            n = A[topa];
            if (topb > 0)
                if (A[topa] > B[topb])
                {
                    cout << "大盘压小盘，非法移动!" << endl;
                    Sleep(900);
                    continue;
                }
        }
        else if (src == 'A' && dst == 'C')
        {
            n = A[topa];
            if (topc > 0)
                if (A[topa] > C[topc])
                {
                    cout << "大盘压小盘，非法移动!" << endl;
                    Sleep(900);
                    continue;
                }
        }
        else if (src == 'B' && dst == 'C')
        {
            n = B[topb];
            if (topc > 0)
                if (B[topb] > C[topc])
                {
                    cout << "大盘压小盘，非法移动!" << endl;
                    Sleep(900);
                    continue;
                }
        }
        else if (src == 'B' && dst == 'A')
        {
            n = B[topb];;
            if (topa > 0)
                if (B[topb] > A[topa])
                {
                    cout << "大盘压小盘，非法移动!" << endl;
                    Sleep(900);
                    continue;
                }
        }
```

```
        else if (src == 'B' && dst == 'A')
        {
            n = B[topb];;
            if (topa > 0)
                if (B[topb] > A[topa])
                {
                    cout << "大盘压小盘，非法移动!" << endl;
                    Sleep(900);
                    continue;
                }
        }
        else if (src == 'C' && dst == 'A')
        {
            n = C[topc];
            if (topa > 0)
                if (C[topc] > A[topa])
                {
                    cout << "大盘压小盘，非法移动!" << endl;
                    Sleep(900);
                    continue;
                }
        }
        else if (src == 'C'&& topc>0 && dst == 'B')
        {
            n = C[topc];
            if (topb > 0)
                if (C[topc] > B[topb])
                {
                    cout << "大盘压小盘，非法移动!" << endl;
                    Sleep(900);
                    continue;
                }
        }
        disk_move(n, src, dst, op);
        move(n, src, dst);
        print_tower();
        cct_gotoxy(0, 17 + 20);
        cout << "第" << setw(4) << cnt << "步" << "(" << n << "#: " << src << "—>" << dst << ")  ";
        print_array();
        if (checkend(end, end_char))
        {
            cct_gotoxy(0, 20 + 17 + 1 + 1);
            cout << "游戏结束!!!!!" << endl;
            break;
        }
        pause();
    }
```