汉诺塔

综合实验报告

学号: 2351050

姓名: 杨瑞晨

班级: 信05/02班

完成日期: 24/05/14

1. 题目分析

1.1 题目综述

汉诺塔问题是一个经典数学问题。有三根杆子，初始杆上有 N 个圆盘，尺寸由下到上依次变小。要求每次只能移动一个圆盘，大盘不能叠在小盘上面，将所有圆盘移至目标杆。

在之前的作业中，我们已经实现了汉诺塔的初步解决。现要求将其各种实现方式集成于一个程序，通过菜单进行选择，并通过 cmd 窗口进行伪图形界面展示。

1.2 功能分析

菜单项共有10个，分别完成以下功能：

1) 基本解，对应 4-b7；
2) 基本解-添加步数记录，对应 4-b8；
3) 横向显示汉诺塔内部数组，对应 5-b6；
4) 横向与纵向显示汉诺塔内部数组，对应 5-b7；
5) 初始化图形界面-画出三根空柱子；
6) 初始化图形界面-画出 n 个盘子；
7) 实现第 1 个盘子的移动；
8) 实现汉诺塔问题的图形解；
9) 实现人工操作汉诺塔移动，即最终游戏版本。

1.3 要求与限制

文件由6个文件组成，2个已给出(斜体标注)：

cmd_console_tools.h
hanoi.h—头文件，用于引入其他头文件及宏定义，以及声明函数
cmd_console_tools.cpp
hanoi_main.cpp—main函数
hanoi_menu.cpp—显示菜单项及选择
hanoi_multiple_solutions.cpp—菜单中各项功能的实现

仅以下内容允许使用全局变量（全局const变量/宏定义不受限）：

- 总移动步数： 1个全局简单变量
- 圆柱现有圆盘编号： 3个全局一维数组/1个全局二维数组
- 圆柱现有圆盘数量： 3个全局简单变量/1个全局一维数组
- 延时： 1个全局简单变量

其他要求：

- 只可使用一次cct_cls()命令清屏，其余只允许擦除现有位置；
- 只允许使用1个共用递归函数（1/2/3/4/8），用参数解决不同要求间的差异，不超过15行；
- 处理输入（1/2/3/4/6/7/8）须共用同一函数，仅此函数允许使用函数形参为实参的指针；
- 内部数组输出（横向 3/4/8；纵向 4/8）共用一个函数
- 图形处理（画柱子、盘子、盘子移动）共用一个函数

2. 整体设计思路

2.1 基本解

汉诺塔问题可以使用递归算法解决（A-->C）：

1) 将 A 柱顶部的 n-1 块盘移动到 B 柱；
2) 将 A 柱剩下的大盘移动到 C 柱；
3) 将 B 柱的 n-1 块盘移动到 C 柱。

2.2 问题的解决思路

为便于编程求解汉诺塔问题，可将汉诺塔中每个柱子抽象为一个一维数组，分别对应 stackA[], stackB[], stackC[] ；将数组中的元素对应不同大小的盘子。开始时进行数组初始化将初始柱上的“盘子”填入。

盘子的移动便可抽象为数组中元素的更替。将数组理解为栈，移动便可理解为数组元素的出栈、入栈。引入栈顶指针 topA, topB, topC 用于指向数组顶的上一个元素。引入中间变量 temp 作数字的存储。则移动过程即为：

```
temp = stackN[--topN];
stackN[topN++] = temp;
```

2.3 程序实现思路

2.3.1 头文件

头文件用于以下几个作用：

1) 引入其他头文件；
2) 定义 #define 宏定义及 const 型常变量；
3) 存放函数声明以便不同 cpp 文件间的互相访问；

2.3.2 main

main 文件用于初始化屏幕、调用菜单函数并返回菜单项以根据选项调用 solutions 中的 hanoiFinal 函数以完成不同的效果。

2.3.3 menu

menu 文件用于显示菜单并返回选项。

2.3.4 multiple Solutions

根据 menu 的返回值调用各菜单项实现所需函数。主要有以下几个功能：

1） 利用递归算法解决汉诺塔问题；

2） 利用栈完成汉诺塔内部数组元素的移动及内部横向/纵向数组的显示；

3） 图形化相关函数，完成绘图及图形的移动；

4） 根据菜单项调用以上部分的不同函数完成对应输出。

2.3.5 cmd_console_tools

cct_ 系列函数，用于清屏、光标移动、图形化界面等。

3. 主要功能的实现

3.1 利用递归算法解决汉诺塔问题

设起始柱为 src ，目标柱为 dst ，中间柱为 tmp。

设共有 n 个盘子，每一步便可视作将前 n-1 个盘子由起始柱移至中间柱，第 n 个盘子移至目标柱。当 n>1 时调用函数 hanoi(n-1)； n=1 时输出每一步移动过程。

可表示为：

```
void hanoi(n, src, tmp, dst)
{
    if (n == 1) {
        move(n, src, dst);
    }
    else {
        hanoi(n - 1, src, dst, tmp); //将 n-1 块盘子由起始柱移至中间柱
        move(n, src, dst);
        hanoi(n - 1, tmp, src, dst); //将 n-1 块盘子由中间柱移回起始柱
    }
}
```

3.2 利用数组和栈方式表示柱子上盘子的情况并实现移动

3.2.1 初始化

- 定义 3 个全局一维数组 A[10], B[10], C[10] 用于存放盘子编号。

- 定义 3 个全局简单变量 topA, topB, topC 用于表示栈顶位置指向元素待插入位置（当前元素的上一个）。=
- 使用 initial 函数遍历数组将 src 柱中充填初始状况。如：
  起始柱为 A ，共 4 层，调用函数使 stackA[10]={4,3,2,1,0,..,0}，stackB 与 stackC 中元素全为0；topA 为 5 ， topB 与 topC 为 0.
- 使用 clearStack 函数清除当前栈中所有数字，便于下次重新使用。

3.2.2 盘子移动

盘子的移动通过数组对应元素的出栈与入栈实现，如：

起始柱为 A ，目标柱为 C. A 的栈顶指针自减，栈顶元素出栈存入中间变量 temp ,；元素进入 C ，中间变量赋值给当前指针处，栈顶指针自加。

```
temp = stackA[--topA];
stackC[topC++] = temp;
```

3.2.3 根据菜单项选择调用不同分支

- 由于菜单项选择的不同，所完成的效果亦不相同，故使用 printMove 函数，根据菜单项的不同，执行不同的语句，并在 hanoi 递归函数中调用。
- 在分支 3/4/8/9 中调用 printLevelStack 打印内部横向数组；
- 在分支 4/8/9 中调用 printVerticalStack 打印内部纵向数组；
- 在分支 5/6/7/8/9 中调用 drawColumn 绘制柱子；
- 在分支 6/7/8/9 中调用 drawPlate 绘制盘子；
- 在分支 7/8/9 中调用 plateMoveVertical 完成盘子的垂直移动，调用 plateMoveLevel 完成盘子的水平移动；

3.3 其他

3.3.1 输入函数

input 函数用于输入，形参为指针型变量，便于改变多个实参的值。

```
void input(int *n, char *src, char *tmp, char *dst, char select); //形参为指针型变量
```

3.3.2 绘图函数

使用 cct_ 系列函数完成绘图及光标移动操作。

- cct_gotoxy(X,Y): 使光标移动到 (X,Y)；
- cct_setcursor(CURSOR_INVISIBLE): 将光标设置为隐藏状态；
- cct_showch(X,Y,’ ’,bg_color,fg_color,length): 在 (X,Y)初绘制长度为 length ，背景色为 bg_color ， 前景色为 fg_color 的‘ ’字符；
- cct_setcolor(): 将颜色设置重新调为默认值，以保证后续的正常输出。

盘子的移动可以精简为以上函数的组合。擦除（cct_showch）原位置盘子，移动（cct_gotoxy）至下一位置，绘制（cct_showch）盘子,擦除过程中应注意重新补全柱子的位置。

上移次数根据起始柱栈顶指针决定，下移次数由目标数栈顶指针决定；平移次数由起始柱与目标柱相对位置决定。

3.3.3 延时函数

根据用户输入的全局变量 sleepTime 的值完成不同的延时功能。

- 0 调用 _getch()函数读取输入的回车完成回车执行下一步的效果。
- 1-5 调用 Sleep 函数，实现不同的延时效果，分别为延时200，100，50，10，不延时。

3.3.4 游戏版本的特殊处理

使用 cin.getline() 函数处理输入，使用字符数组 t[] 用于存储所读取的字符。若 t[0]为Q/q时中止程序，t[0] t[1] 为合规的字符时(并进行大小写转换)进行移动（调用全局栈）；

若发生以下两种错误状况时进行提示：

1. 原柱为空。通过栈顶指针判断，若栈顶指针为 0 ，说明此时柱上并无圆盘；
2. 大盘压小盘。通过栈顶指针判断，若起始柱指针值大于目标柱指针值，则说明移动后会发生大盘压小盘现象。

最后，使用 checkFinish() 函数对游戏结束进行判定，若判定成功，输出“游戏结束!”。

4. 调试过程碰到的问题

4.1 菜单项 7 - 盘子的第 1 次移动

由于盘子的第一次移动不一定是由起始柱移向目标柱，也有可能是移向中间柱，而在实现过程发现会发生移动方向错误，移动次数不正确等问题，使人百思不得其解。

在 Visual Studio 调试功能的帮助下，通过设置断点等操作，观察函数实时调用参数，发现即使将 n 设置为 1，仍然会不断调用递归函数，得到错误的结果。

最终，通过在递归函数中加入若菜单项为 7 时中止调用（return;）递归函数，最终取得正确的第 1 个盘子的参数完成移动。

4.2 菜单项 9 - 控制移动功能的读取

在起初的设想中，使用 cin 函数读取，会发生以下问题：

- 无法正确读取想要的字符，若处理为仅读取到 A B C a b c 输出至屏幕并存入，会导致后续输入缓冲区中还有其它字符，且会发生顺序问题，无法读到想要的字符；
- 不便于清除缓冲区，每次输入的正确与否与输入的长度均未知。

后改用 cin.getline() 函数读取数据并存入字符数组，解决了上述问题。

5. 心得体会

5.1 心得体会与经验总结

- 善于使用编译器的 Debug 功能（如设置断点）及输出中间变量的方式，构建临时测试，减轻后续调试过程中寻找 bug 的困难程度；
- 多次出现的相似代码应思考将其集成为函数，减少代码量，便于理解；
- 反复出现的常量应使用 const 常变量/#define 宏定义的方式写于头文件，便于后续维护；
- 写代码前充分思考，避免花费大量时间却得到了无用的代码；
- 合理注释，便于后期维护时自己和他人对代码的理解。

5.2 对复杂问题编写的体会

- 复杂问题往往组成庞杂，开始写之前须构建好主体的思路，否则无从下手。也不要想到哪就写到哪，不利于整体结构的构建；
- 善于利用不同的 cpp 文件化繁为简，善于使用头文件完成函数声明及宏定义与常变量的声明；
- 将问题模块化，分解为不同的部分，依据主要功能分成大的模块，再拆分成小的模块，最终组合在一起实现复杂功能；
- 遇到思路卡壳时，不妨绘制流程图以更好理解当前问题。

5.3 对各题前后关系及代码的重新利用

此次作业便是对各题前后联系与代码的有效重新利用的最好证明，有效减少了工作量。之前的问题循序渐进，逐渐深入，循循善诱。菜单项 1-4 均可从之前做过的作业中找到，在后续的使用中只需适当增添参数即可，比如根据不同的菜单项完成些许不同的输出。合理的代码复用，可大大减轻复杂程序的编写难度。

5.4 如何更好的使用函数

- 好的函数应各司其职，复用性强。善于提炼代码中的共性，写成具体的函数。每个函数尽可能实现一个具体的功能，而非糅杂起来的臃肿函数，以便于后续需要实现相似功能时可以重复利用；
- 在编写函数时，为方便后续的使用，善于预留接口和参数十分重要，便于应对以后过程中出现新的要求与问题；
- 复杂功能应通过有条理的调用不同函数完成复杂功能，以大大减少代码重复量，精简程序。

6. 附件：源程序

6.1 hanoi-main

```
int main()
{
    cct_setconsoleborder(120, 40, 120, 9000);
    cct_setfontsize("新宋体", 24);
    while (1) {
        const int select = menu();
        cout << endl;
        if (select == '0')
            return 0;
        hanoiFinal(select);
        cout << endl << "按回车键继续" << endl;
        while (1) {
            int enter = _getch();
            if (enter == 13) {
                break;
            }
        }
    }
    return 0;
}
```

6.2 hanoi_menu

```
char menu()
{
    cct_setconsoleborder(120, 40, 120, 9000);
    cct_cls();
    //此处省略菜单项
    char ret;
    //错误处理
    cout << ret << endl;
    return ret;
}
```

6.3 hanoi_multiple_solutions

6.3.1 全局变量定义

```
int topA = 0, topB = 0, topC = 0;
int stackA[10] = { 0,0,0,0,0,0,0,0,0,0 };
int stackB[10] = { 0,0,0,0,0,0,0,0,0,0 };
int stackC[10] = { 0,0,0,0,0,0,0,0,0,0 };
int countN = 0;
int sleepTime = 3;
```

6.3.2 延迟函数

```
void delay()//延迟时间
{
    switch (sleepTime) {
        case 0:
            while (1) {
                int enter = _getch();
                if (enter == 13)
                    break;
            }
            break;
        case 1:
            Sleep(200);
            break;
        //…
        case 5:
            break;
        default:
            break;
    }
}
```

6.3.3 移动与水平/垂直数组的打印函数

```
//省略 B C
void move(char src, char dst)//栈中数字移动
{
    int temp = 0;
    if (src == 'A') {
        temp = stackA[--topA];
        stackA[topA] = 0;
    }
//…
    if (dst == 'A') {
        stackA[topA++] = temp;
    }
//…
}
void printLevelStack()//打印水平内部数组
{
    int i;
    cout << " A:";
    for (i = 0; i < 10; i++) {
        if (stackA[i] == 0)
            cout << setw(2) << " ";
        else
            cout << setw(2) << stackA[i];
    }
    //…
}
void printVerticalStack(char select) //打印垂直内部数组
{
```

```
    const int X = 28;
    int Y;
    if (select == '8' || select == '9')
        Y = 28;
    else
        Y = 15;
    cct_gotoxy(X - 1, Y + 1);
    cout << setw(25) << setfill('=') << "=" <<
endl;
    cct_gotoxy(X + 1, Y + 2);
    cout << setfill(' ') << "A         B
C";
    int i;
    for (i = 0; i < 10; i++) {
        cct_gotoxy(X, Y - i);
        if (stackA[i] == 0)
            cout << setw(2) << " ";
        else
            cout << setw(2) << stackA[i];
    }
    //…
}
void clearStack()//清空栈及计数器
{
    countN = 0;
    sleepTime = 3;
    for (int i = 0; i < 10; i++) {
        stackA[i] = 0;
        //…
    }
    topA = 0;
    //…
}
void initial(int n, char src)
{
    //初始化栈
    int i = n, j = 0;
    for (i = n; i > 0; i--) {
        if (src == 'A')
            stackA[j] = i;
        //…
        j++;
    }
    if (src == 'A')
        topA = j;
    //…
}
```

6.3.4 输入函数

```
void input(int *n, char *src, char *tmp, char
*dst, char select) //形参为指针型变量
{
    while (1) {
        cout << "请输入汉诺塔的层数(1-10)" <<
endl;
        cin >> *n;
        //输入错误处理
    }
    while (1) {
        cout << "请输入起始柱(A-C)" << endl;
        cin >> *src;
        //输入错误处理
    }
    while (1) {
        cout << "请输入目标柱(A-C)" << endl;
        cin >> *dst;
        //输入错误处理
        }
    }
    if (select == '4' || select == '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 >> sleepTime;
            //输入错误处理
        }
    }
    *tmp = 3 * 'B' - *src - *dst;
    initial(*n, *src);
}
```

6.3.5 hanoi 递归函数

```
void hanoi(int n, char src, char tmp, char dst,
char select)//共用的递归函数
{
    if (n == 1) {
        printMove(n, src, dst, select);
    }
    else {
        hanoi(n - 1, src, dst, tmp, select);
        if (select == '7')//7 只有第一个盘移动
            return;
        printMove(n, src, dst, select);
        hanoi(n - 1, tmp, src, dst, select);
    }
}
```

6.3.6 菜单项调用函数

```
void printMove(int n, char src, char dst, char
select) //打印移动
{
    const int X = 0;
    const int Y = 20;
    countN++;
    if (select == '1')
        //文字 表示移动
    else if (select == '2')
        //文字 表示移动
```

```
    else if (select == '3') {
        //文字 表示移动
        move(src, dst);
        printLevelStack();
        cout << endl;
    }
    else if (select == '4') {
        delay();
        cct_gotoxy(X, Y);
        //文字 表示移动
        move(src, dst);
        printLevelStack();
        printVerticalStack(select);
    }
    else if (select == '6') {
        //文字 表示移动
    }
    else if (select == '7') {
        move(src, dst);
        plateMoveVertical(n, src, dst, UP);
        plateMoveLevel(n, src, dst);
        plateMoveVertical(n, src, dst, DOWN);
    }
    else if (select == '8') {
        cct_setcolor();
        delay();
        cct_gotoxy(0, 32);
        //文字 表示移动
        move(src, dst);
        printLevelStack();
        printVerticalStack(select);
        plateMoveVertical(n, src, dst, UP);
        plateMoveLevel(n, src, dst);
        plateMoveVertical(n, src, dst, DOWN);
    }
    else if (select == '9') {
        cct_setcolor();
        cct_gotoxy(0, 32);
        //文字 表示移动
        move(src, dst);
        printLevelStack();
        printVerticalStack(select);
        plateMoveVertical(n, src, dst, UP);
        plateMoveLevel(n, src, dst);
        plateMoveVertical(n, src, dst, DOWN);
    }
}
void hanoiFinal(char select)//最终的函数，在菜单
中调用
{
    int n;
    char src, dst, tmp;
    if (select != '5') {
        input(&n, &src, &tmp, &dst, select);
    }
    if (select == '1' || select == '2' || select
== '3' || select == '4') {
        if (select == '4') {
            cct_cls();
            //文字 表示移动
            cct_gotoxy(0, 20);
            cout << "初始:                ";
            printLevelStack();
            printVerticalStack(select);
        }
        hanoi(n, src, tmp, dst, select);
        if (select == '4')
            cct_gotoxy(0, 24);
    }
    else if (select == '5') {
        cct_cls();
        drawColumn();
        cct_gotoxy(0, 24);
    }
    else if (select == '6') {
        cct_cls();
        printMove(n, src, dst, select);
        drawColumn();
        drawPlate(n, src);
        cct_gotoxy(0, 24);
    }
    else if (select == '7') {
        cct_cls();
        cct_gotoxy(0, 0);
        cct_setcolor();
        //文字 表示移动
        drawColumn();
        drawPlate(n, src);
        hanoi(n, src, tmp, dst, select);
        cct_gotoxy(0, 24);
    }
    else if (select == '8') {
        cct_cls();
        cct_gotoxy(0, 0);
        cct_setcolor();
        //文字 表示移动
        cct_gotoxy(0, 32);
        cout << "初始:                ";
        printLevelStack();
        printVerticalStack(select);
        drawColumn();
        drawPlate(n, src);
        hanoi(n, src, tmp, dst, select);
        cct_gotoxy(0, 36);
    }
    else if (select == '9') {
        cct_cls();
        cct_gotoxy(0, 0);
        cct_setcolor();
        //文字 表示移动
        cct_gotoxy(0, 32);
        cout << "初始:                ";
        printLevelStack();
        printVerticalStack(select);
```

```
        drawColumn();
        drawPlate(n, src);
        game(n, src, dst, select);
        cct_gotoxy(0, 36);
    }
    cct_setcursor(CURSOR_VISIBLE_NORMAL);
    cct_setcolor();
    clearStack();
}
```

6.3.7 绘图函数

```
void drawColumn()//画柱子
{
    cct_setcursor(CURSOR_INVISIBLE);
    const int X = 2;
    const int Y = 16;
    int LENGTH = 23;
    const int bg_color = COLOR_HYELLOW;  //背景为亮黄色
    const int fg_color = COLOR_HYELLOW;
    for (int i = 0; i < 3; i++) { //画横向底座
        cct_showch(X + i * (LENGTH + 3), Y, ' ', bg_color, fg_color, LENGTH);
        Sleep(50);
        for (int j = 0; j < 12; j++) //画纵向轴
        {
            cct_showch(X + 11 + i * (LENGTH + 3), Y - j, ' ', bg_color, fg_color, 1);
            Sleep(50);
        }
    }
}
void drawPlate(int n, char src)//画初始盘子
{
    cct_setcursor(CURSOR_INVISIBLE);
    const int X = 2;
    const int Y = 16;
    int LENGTH = 23;
    for (int i = n; i > 0; i--) {
        Sleep(80);
        cct_showch(X + 11 - i + (src - 'A') * (LENGTH + 3), Y - 1 - n + i, ' ', i, i, i * 2 + 1);
    }
}
void plateMoveVertical(int n, char src, char dst, bool direction)//盘子垂直移动    direction: UP -- 1  DOWN -- 0
{
    cct_setcursor(CURSOR_INVISIBLE);
    const int X = 2;
    const int Y = 16;
    const int endY = 2;
    int LENGTH = 23;
    if (direction) {
        if (src == 'A') {
            for (int y = Y - 1 - topA; y > endY; y--) {
                if (sleepTime != 0) {
                    delay();
                }
                else
                    Sleep(50);
                cct_showch(X + 11 - n + (src - 'A') * (LENGTH + 3), y, ' ', COLOR_BLACK, COLOR_WHITE, n * 2 + 1);//擦除原有的盘子
                if (y > 4) {
                    cct_showch(X + 11 + (src - 'A') * (LENGTH + 3), y, ' ', COLOR_HYELLOW, n, 1);//补齐亮黄色柱子
                }
                if (sleepTime != 0) {
                    delay();
                }
                else
                    Sleep(50);
                cct_showch(X + 11 - n + (src - 'A') * (LENGTH + 3), y - 1, ' ', n, n, n * 2 + 1);//打印新的盘子
            }
        }
        //…
    }
    else {
        if (dst == 'A') {
            for (int y = endY; y <= Y - 1 - topA; y++) {
                if (sleepTime != 0) {
                    delay();
                }
                else
                    Sleep(50);
                cct_showch(X + 11 - n + (dst - 'A') * (LENGTH + 3), y, ' ', COLOR_BLACK, COLOR_WHITE, n * 2 + 1);
                if (y > 4) {
                    cct_showch(X + 11 + (dst - 'A') * (LENGTH + 3), y, ' ', COLOR_HYELLOW, COLOR_HYELLOW, 1);
                }
                if (sleepTime != 0) {
                    delay();
                }
                else
                    Sleep(50);
                cct_showch(X + 11 - n + (dst - 'A') * (LENGTH + 3), y + 1, ' ', n, n, n * 2 + 1);

            }
        }
        //…
```

```
    }
}
void plateMoveLevel(int n, char src, char dst)//盘子水平移动
{
    cct_setcursor(CURSOR_INVISIBLE);
    const int X = 2;
    const int Y = 16;
    const int endY = 2;
    int LENGTH = 23;
    int DIRECTION = (dst - src) > 0 ? 1 : -1; //判断移动方向    1--右移  -1--左移
    int x = X + 11 - n + (src - 'A') * (LENGTH + 3);
    while (1) {
        if (sleepTime != 0) {
            delay();
        }
        else
            Sleep(50);
        cct_showch(x, endY, ' ', COLOR_BLACK, COLOR_WHITE, n * 2 + 1);//擦除原盘
        if (sleepTime != 0) {
            delay();
        }
        else
            Sleep(50);
        if (x == X + 11 - n + (dst - 'A') * (LENGTH + 3))
            break;
        cct_showch(x + DIRECTION, endY, ' ', n, n, n * 2 + 1);//打印新盘
        x += DIRECTION;
    }
}
```

6.3.8 对游戏 9 特殊处理的函数

```
void game(int n, char src, char dst, char select)//对9特殊处理的函数
{
    char t[100];
    while (1) {
        int from = 1024, to = 1024;
        cct_gotoxy(0, 34);
        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(60, 34);
        cin.getline(t, 100);
        if (t[0] == 'Q' || t[0] == 'q') {//游戏中止
            cout << "游戏中止!          ";
            return;
        }
    //转大写
        if (t[0] >= 'A' && t[0] <= 'C' && t[1] >= 'A' && t[1] <= 'C') {
            if (t[0] == 'A' && topA > 0)//给from填入当前栈的值
                from = stackA[topA - 1];
            //…
            else {
                cct_gotoxy(0, 35);
                cct_setcolor();
                cout << "原柱为空!          ";
                continue;
            }
            if (t[1] == 'A' && topA > 0)
                to = stackA[topA - 1];
            //…
            if (to > from) {
                cct_gotoxy(0, 35);
                cct_setcolor();
                cout << "                    ";
                printMove(from, t[0], t[1], select);//从t[0]移动到t[1],大小为from
            }
            else {
                cct_gotoxy(0, 35);
                cct_setcolor();
                cout << "大盘压小盘，非法移动!              ";
                continue;
            }
            if (checkFinish(n, dst)) {
                cct_gotoxy(0, 35);
                cct_setcolor();
                cout << "游戏结束!          ";
                return;
            }
        }
    }
}
bool checkFinish(int n, char dst)//游戏结束判定
{
    int temp = n;
    if (dst == 'A') {
        for (int i = 0; i < n; i++) {
            if (stackA[i] == temp)
                temp--;
            else
                return 0;
        }
        return 1;
    }
    //…
    return 0;
}
```