同济大学 实验报告

Hanoi伪UI版解题报告

班级：计算机一班

姓名：王哲源

学号：1652228

装 订 线

完成日期：2016.12.13

装 订 线

1.题目及其描述

题目主要目的在于解决经典汉诺塔问题，同时展示出具体操作过程以及 CMD 伪图形界面，最终在此基础上制作出基于以上功能的简易汉诺塔游戏。其一共包含以下 10 种操作

1.1 基本解

逐步展示每一步将哪一个盘子如何移动

1.2 基本解（步数记录）

在 1.1 的基础上加入当前为第几步的步数统计

1.3 内部数组显示（横向）

在 1.3 的基础上加入以横向数组记录的各个柱子在某一步操作之后的状态

1.4 内部数组显示（横向+纵向）

在 1.4 的基础上加入以纵向数组表示的栈模型，同时打印方式由一次性打印变为可设置延时/手动控制的逐步打印

1.5 图形解-预备-画三个柱子

为伪图形界面实现的预备工作，通过 cmd 背景颜色调整绘制三个柱子的伪图形界面

1.6 图形解-预备-在起始柱上画 n 个盘子

为伪图形界面实现的预备工作，在 1.5 的基础上绘制出起始的伪图形状态

1.7 图形解-预备-第一次移动

为伪图形界面实现的预备工作，在 1.6 的基础上实现以伪图形界面展示的汉诺塔第一步操作

1.8 图形解-自动移动版本

基于 1.7 及 1.4 而实现的汉诺塔伪图形界面版本

1.9 图形解游戏版

将 1.8 中的自动移动变为可由用户手动输入而操作的游戏版，同时支持非法输入及非法移动的判断

1.10 退出

2. 整体设计思路

根据题目要求，对程序进行分块，主要包含以下 8 个部分

2.1 main part

主函数部分，用于实现程序的基础部分，区分用户选择模式及各函数调用

2.2 screen operate part

CMD 窗口操作部分，用于输出简易 UI 界面以及调整 CMD 窗口大小

2.3 input part

读入部分，通过用户在 CMD 窗口键入的数据读入盘子个数及起始和终止柱

2.4 game part

游戏部分，编写游戏版操作部分

2.5 work part

主操作部分，包括初始化以及递归求解函数

2.6 stack operate

栈操作部分，包含模拟进栈及出栈操作

2.7 output part

输出部分，对除伪图形界面外其他各种输出方式进行输出

2.8 img part

伪图形界面输出部分，实现柱子绘制、盘子移动的伪图形界面输出

3. 主要功能实现

以下对 2 中提到的各个部分主要功能实现进行较详细的解释

3.1 main part

主函数部分以一个永真表达式作为循环语句的判断部分，实现程序多次操作，每当循环执行，先对 output_system 函数进行调用复原 cmd 窗口大小并输出模式选择界面，再通过调用 init()函数对题目所需参数进行读入，以 init 函数返回值判断用户是否输入 0 要求退出程序，若非 0 则再通过用户输入选项调用对应函数实现各类操作

3.2 screen operate part

change_screen 函数根据传入的用户选择模式调整 cmd 窗口的大小

3.3 input part

输入部分以 int 类型的 init 函数为主体，先对 init_mode 函数进行调用，读入用户选择模式 mode，根据模式判断是否合法，若不合法则返回主函数重新执行循环；若合法则判断是否为 0，为 0 则范围值 0，从主函数退出程序，否则根据 mode 具体值分别调用 init_n ,init_st ,init_ed ,init_speed 函数分别对盘子个数、起始及终止柱，显示延时进行读入，最后调用 pre_work 函数处理出中间柱

3.3.1 pre_work 函数

传入 st 和 ed 两个参数，以一个大小为 3 的 bool 标志数组对 st 和 ed 进行标记，最终未被标记的即为中间柱

3.3.2 输入合法性判断

以永真表达式作为循环判断条件置于读入函数外，通过 scanf 函数返回值判断读入的数据是否合法，若不合法则继续执行循环重新读入，若合法判断其是否在定义域内，若不在则继续执行循环重新读入

3.3.3 读入缓冲区的清空及多余字符的清除

通过 FFLUSH 操作（实际为 define 的一个 while(getchar()!='\n')语句）不停读入非法/多余输入数据直到回车为止，保证下一次读入的正确性

3.4 game part

主体为 game_ver 函数，其中含有一个以永真表达式作为判断语句的循环，每次循环开头调用 check 函数确定游戏是否已操作完成。若游戏未结束，则先调用 clear()函数，通过调整光标未知并输出空格覆盖提示输入语句及其下一行放置上一步操作语句残留，之后再将提示输入语句重新打印，并通过用于输入判断/执行用户指定的移动操作

3.4.1 游戏结束的判断

此处无需对终止柱对应栈判断是否包含从 1-n 盘子，只需要询问 top[ed-‘A’]是否为 n 即可确定游戏是否结束

3.4.2 move_operate 函数

该函数用于实现柱子的移动，先对来源柱的 top 进行判断是否为空，再取来源柱及目标柱的栈顶盘编号通过对比确定是否为合法移动操作。

*此处应注意在 main 函数初始化中对 stack[i][0](i=1,2,3)进行赋值，值为 n+1，保证其始终为最大的柱子以防柱为空时无法移动

3.5 work part

这部分函数是由之前作业复制过来的。主体是 move 函数，通过递归的方式调用 move 函数，模拟汉诺塔的移动方式，并每当执行一步对答案按照格式要求输出。

3.5.1 汉诺塔递归求解

由于汉诺塔的求解是由若干个子问题，即对于当前的某一个盘子，我应该如何移动而组成的，所以我们不妨设某一步（*注意，不是指整个过程，而是某一子过程）的目的是从起始柱(A)移动目标柱为 C，其中默认盘子从小到大编号为 1-n。

则当 n=1 时只需要将 1 号盘从 A 移动到 C 即可

当 n=2 时，我们需要先讲 1 移动到 B，再将 2 移动到 C,最后再把 1 从 B 移动到 C

那么我们可以这么考虑，如果我们要把当前第 n 个盘子移动到 C，则需要先将上面第(n-1)号盘子借助目标柱(C)移动到中间柱(B)，才能将 n 从 A 移动到 C，最终将第(n-1)号盘（此时位于中间柱上）借助 起始柱子(A)移动目标柱(C)

*边界条件：当且仅当 n=1 时，我们不需要借助中间柱，只需要直接把盘子由起始柱移动到目标柱即可

通过调用这个递归过程，便可以实现将 A 上的 n 个盘子移动到 C 上。同时由于每次递归中都是执行一个操作，即将当前第 n 号盘子由当前步骤的起始柱移动到目标柱，所以我们只需要在递归函数中每次调用一次输出操作即可

3.6 stack operate

包含 push 和 pop 两个函数，push 包含两个参数 x,i 表示将 x 元素放入 i 号栈中；pop 函数包含一个参数 i 表示弹出第 i 号栈栈顶元素，同时有一个 int 返回值返回弹出元素的编号便于实现盘子的移动

3.6.1 关于栈的存储

以一个全局一维数组 top[3]及一个全局二维数组 stack[3][Maxn]分别表示 A,B,C 三个栈的当前栈中元素个数及栈中状态，top 始终指向栈顶（栈下标由 1 开始），并通过（柱子编号-‘A’）对应栈的编号

3.6.2 关于栈的清空

由于 stack 数组中由 top 控制其合法区间即可，本没有必要在函数出栈时对栈进行清空。但由于输出方式的问题（3.7.2 中将会提及），因此对于栈的复原仍然十分重要。否则更有可能导致 BUG(4.3 中会提及)。

3.7 output part

此部分包含了除图形界面外其余所有情况的输出，通过传入 mode 参数控制输出格式

3.7.1 堆栈形式的输出部分

这部分是由之前作业复制过来的。由于当时写作业时暂未获得通过控制光标位置输出的许可，所以采用了比较原始的逐行打印方法。由于汉诺塔高度不超过 10，且堆栈本身也是自上而下的存储方式，因此每次遍历三个栈，若栈中元素不为 0，则将其打印，由此实现堆栈形式的输出

3.7.2 逐次输出相关

由于最初未使用控制光标坐标的方法，为了不对原有程序进行过多修改，此处采用输出一次后对输出部分清屏的方式，然后再将光标移动至头部重新打印。

同时注意到有/无伪图形界面的数组输出位置不同，因此为了避免把伪图形界面一同清空，此处传入 mode 参数根据具体情况选择清屏起始坐标。

3.8 img part

伪图形界面输出部分主要包括三个，柱子生成，盘子生成，移动

3.8.1 柱子生成

采用移动光标的方法，先横向打印柱子底部，再纵向由下往上打印柱子部分。此处可以采用一个全局变量存储柱底位置,再用一个大小为 3 的全局数组存储三根柱子中心位置（后面会用到）

3.8.2 盘子生成

采用移动光标的方法，纵向坐标以柱子底部为准，自下而上打印盘子，底色以盘子编号 1-n 为背景色，便于之后盘子移动修改。横向起始以柱子中轴坐标为对称轴，2*i+1 为盘子长度输出即可

3.8.2 盘子移动

该部分为最繁琐部分，由于伪图形界面通过修改底色以达到图形化，因此每次移动盘子都需要用原背景色（黑色）覆盖盘子上一次位置，再打印盘子移动后新位置。同时由于颜色覆盖会将柱子一同覆盖，因此使用 3.8.1 中记录的柱子中轴线便可以快速查询柱子坐标。同时，使用一个值为 1/-1 的变量作为加法元，其值由目标柱相对起始柱的位置决定，即可避免使用两个循环语

句

4.调试过程遇到的问题

调试过程主要错误集中于读入与输出部分，以下选取 3 个主要问题进行概述

4.1 读入错误的处理

原本采用的是 fflush(stdin)的方式,发现当读入 int 类型输入 char 类型会发生死循环问题，疑似 fflush(stdin)刷新读入缓冲区并非清空读入缓冲区。改用 cin.ignore(1024,'\n'); cin.clear();发现仍然是死循环。

后来了解要先调用 cin.clear()清除 cin 中的不正确标志，cin.ignore()调用才有意义。

但为了不对程序进行太大的改动,最终还是采用了 while(getchar()!='\n')不断过滤非法字符的方法处理，通过不断读取多余/非法字符直到回车符的方式来手动清空读入缓冲区，从而解决了死循环问题。

4.2 伪图形界面输出时下侧数组输出问题

由于数组输出采用的是逐行打印，导致出现滚屏情况，即每次输出 cmd 窗口会发生下移而非固定在一个位置不断输出。但由于旧版程序限制将逐行打印改变为光标移动输出工程量过大，最终只能采取添加一个 reset_xy 函数对不同 mode 情况下移动光标输出位置，新步骤输出前对数组输出部分使用空格字符覆盖一遍，达到重置屏幕的目的，再重新打印数组输出部分。

同时应当注意到,当使用空格覆盖之后需要将光标移回起始的输出位置,不然会接着打下去。曾经就因为这个原因导致了滚屏的情况。

4.3 栈部分多次执行程序后输出错误

该问题是在程序完成后发现的。当程序多次执行纵向数组输出/图形输出时，发现栈部分输出开始出现错误。

通过调用逐次打印的操作，发现第一次求解没问题，到了第二次求解的第一个操作就会出现上一次移动结束的结果。经推测很快发现由于之前程序只需要实现一次求解即可，导致执行过程中注意到了栈的复原但结束后未对栈进行复原。而根据 3.7.1 提及的输出方法是遍历整个栈将非 0 元素输出，因此在需要使用 stack 函数的 mode 部分起始加入了 clear_stack 函数，将栈中元素全部清零以防影响下一次运行结果。

5.心得体会

对于这次的程序有以下几点感受：

第一，写长代码思路清晰很重要。作业下发的时候花了大概两天思考框架，大概需要哪些部分，每个任务需要哪些函数，哪些任务需要共用哪些函数，这些都需要在头脑中形成一个框架。有了大概的思路之后再开始动工对于写代码的效率和准确率有挺大的提升。当然，写到中间也会

遇到卡壳的情况，不过都是一些细节问题所以稍微冷静下来思考一阵就能解决。因此写代码前的构思是非常必要的。

第二，程序分块很重要。这一点对于思路清晰化有非常大的帮助。多写函数，把功能相近的函数放在一起，能够让代码写起来思路更加清晰，对于 debug 也有非常大的帮助。适当的注解也能提高 debug 时的效率。

第三，早期代码要尽可能考虑多的事。这一点感受最深，当时的代码由于采用逐行打印方法，这次出现区域输出重置时无法使用清屏，为了避免代码推到重来，只能采用局部覆盖，导致输出部分花费了大量时间。

最后最重要一点就是对于多次执行的程序变量的初始化。尤其是 4.3 中提到的错误就是由于全局变量没有重置导致的，如果不是多次运行否则这个 bug 不可能被发现。因此对于多次运行的程序中某些有可能会干扰下一次运行的变量不能为了节省时间效率而不重置，不然会导致一些严重后果。

6.附件：程序（此处仅包含主程序部分）

```
#define _CRT_SECURE_NO_WARNINGS

#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<iomanip>
#include<Windows.h>
#include<conio.h>
#include "cmd_console_tools.h"

#define For(i,l,r) for(int i=l;i<=r;++i)
#define opFor(i,r,l) for(int i=r;i>=l;--i)
#define FFLUSH while(getchar()!='\n')

using namespace std;

const int Maxh = 10;
const int Maxn = Maxh + 10;
const int X[3] = { 12,44,76 };
const int Y = 15;
const int width = 100;

int cnt;
int stack[3][Maxn];
int top[3];

//img part
void reset_xy(const int mode)
{
    const HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);
    if (mode <= 4)
        return;
    else if (mode >= 5 && mode <= 7)
        gotoxy(hout, 0, 28);
    else if (mode == 8)
        gotoxy(hout, 0, 34);
    else //mode==9
        gotoxy(hout, 0, 38);
    setcolor(hout, COLOR_BLACK, COLOR_WHITE);
}
void put_disk(const int i, const int x, const int y, const int len, const int last)
{
    const HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);
    const int fg_color = COLOR_BLACK;
    const int bg_color = i;
    showch(hout, x, y, ' ', bg_color, fg_color, len);
    Sleep(last);
}
void move_disk(const int i, const char from, const int n, const char to, const int m, const int last)
{
    const HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);
    const int sti = from - 'A';
    const int edi = to - 'A';
    const int stx = X[sti] - i;
    const int edx = X[edi] - i;
    const int len = 2 * i + 1;

    opFor(j, Y - n, 2)
    {
        showch(hout, stx, j, ' ', COLOR_BLACK, COLOR_WHITE, len);
```

```
            if (j >= 3)
                showch(hout, X[sti], j, ' ', COLOR_HYELLOW, COLOR_BLACK, 1);
            put_disk(i, stx, j - 1, len, last);
        }

        int ii = stx > edx ? -1 : 1;
        int p = stx;
        while (p != edx)
        {
            showch(hout, p, 1, ' ', COLOR_BLACK, COLOR_WHITE, len);
            put_disk(i, (p += ii), 1, len, last);
        }

        For(j, 1, Y - m - 1)
        {
            showch(hout, edx, j, ' ', COLOR_BLACK, COLOR_WHITE, len);
            if (j >= 3)
                showch(hout, X[edi], j, ' ', COLOR_HYELLOW, COLOR_BLACK, 1);
            put_disk(i, edx, j + 1, len, last);
        }
    }
    void put_pillar(const int last)
    {
        const HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);
        const int fg_color = COLOR_BLACK;
        const int bg_color = COLOR_HYELLOW;
        const int width = 23;
        const int height = 12;

        showch(hout, 1, Y, ' ', bg_color, fg_color, width);
        showch(hout, 33, Y, ' ', bg_color, fg_color, width);
        showch(hout, 65, Y, ' ', bg_color, fg_color, width);
        opFor(i, 3 + height, 3)
        {
            showch(hout, X[0], i, ' ', bg_color, fg_color, 1);
            Sleep(last);
            showch(hout, X[1], i, ' ', bg_color, fg_color, 1);
            Sleep(last);
            showch(hout, X[2], i, ' ', bg_color, fg_color, 1);
            Sleep(last);
        }
    }
    void img_reset(const int n, const char st, const char ed, const int last)
    {
        printf("从 %c 移动到 %c,共 %d 层", st, ed, n);
        put_pillar(last);
        int k = st - 'A';
        For(i, 1, n)
        {
            int x = X[k] - i, y = Y - (n - i + 1), len = 2 * i + 1;
            put_disk(i, x, y, len, last);
        }

    }

    //output part
    void clear_screen(const int mode)
    {
        const HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);
        gotoxy(hout, 0, 0);
        int sx, sy = 0, ex, ey = width-1;
        if (mode == 4)
            sx = 1, ex = 28;
        else
        {
            sx = Y + 1;
            ex = 32;
        }
        For(i, sx, ex)
        {
            For(j, sy, ey)
            {
                gotoxy(hout, j, i);
                printf(" ");
            }
        }

        if (mode == 4)
            gotoxy(hout, 0, 1);
        else
            gotoxy(hout, 0, Y + 1);
    }
    void print_array()
    {
        printf("A:");
        For(i, 1, top[0])
            cout << setw(2) << stack[0][i];
        For(i, 1, 10 - top[0])
            printf("  ");
        printf(" ");

        printf("B:");
        For(i, 1, top[1])
            cout << setw(2) << stack[1][i];
```

```
        For(i, 1, 10 - top[1])
            printf("  ");
        printf(" ");

        printf("C:");
        For(i, 1, top[2])
            cout << setw(2) << stack[2][i];
        For(i, 1, 10 - top[2])
            printf("  ");

        printf("\n");
    }
    inline void put_pic()
    {
        printf("\n");
        opFor(i, Maxh, 1)
        {
            if (stack[0][i])
                cout   <<   setw(12)   <<
stack[0][i];
            else
                printf("           ");
            if (stack[1][i])
                cout   <<   setw(10)   <<
stack[1][i];
            else
                printf("          ");
            if (stack[2][i])
                cout   <<   setw(10)   <<
stack[2][i];
            else
                printf("          ");
            printf("\n");
        }
        printf("
======================\n");
        printf("             A          B
C\n");
        printf("\n\n\n");
    }
    inline void output(const int mode, const
int n, const char st, const char ed, const int
num, const char from, const char to, const int
last)
    {
        if (mode == 4 || mode == 8 || mode ==
9)
        {
            clear_screen(mode);
            put_pic();
        }

        ++cnt;
        if (mode == 1)
            printf("%d#", num);
        else
        {
            printf("第");
            cout << setw(4) << cnt;
            cout << "步(";
            cout << setw(2) << num;
            cout << "):";
        }
        printf(" %c---->%c ", from, to);

        if (mode == 3 || mode == 4 || mode ==
8 || mode == 9)
            print_array();
        else
            printf("\n");

        if (mode == 8 || mode == 9)
            move_disk(num, from, top[from -
'A'] + 1, to, top[to - 'A'], (last ? last / 10 :
30));

        reset_xy(mode);
    }

    //stack operate
    int pop(int i)
    {
        int res = stack[i][top[i]];
        stack[i][top[i]--] = 0;
        return res;
    }
    void push(int x, int i)
    {
        stack[i][++top[i]] = x;
    }

    //work part
    void wait(const int last)
    {
        if (!last)
        {
            char ch;
            while (1)
            {
                ch = _getch();
                if (ch=='\n' || ch=='\r')
                    break;
            }
        }
        else
            Sleep(last);
    }
    inline void move(const int mode, const int
n, const char st, const char ed, const int num,
const char from, const char to, const char tmp,
const int last)
    {
```

```
        if (!num)
            return;
        move(mode, n, st, ed, num - 1, from, tmp, to, last);
        int x = pop(from - 'A');
        push(x, to - 'A');

        if (mode == 4 || mode == 8)
            wait(last);
        output(mode, n, st, ed, num, from, to, last);
        move(mode, n, st, ed, num - 1, tmp, to, from, last);
    }
    inline void clear_stack(const int n)
    {
        top[0] = top[1] = top[2] = 0;
        stack[0][0] = stack[1][0] = stack[2][0] = n + 1;
        For(i, 1, Maxh)
            stack[0][i] = stack[1][i] = stack[2][i] = 0;
    }
    void reset(const int mode, const int n, const char st, const char ed, const int speed, const int last)
    {
        const HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);

        clear_stack(n);
        int Id = st - 'A';
        opFor(i, n, 1)
            push(i, Id);

        if (mode != 1 && mode != 2)
            system("cls");
        if (mode == 4 || mode == 8)
            printf("从 %c 移动到 %c，共 %d 层，延时设置为 %d ", st, ed, n, speed);
        else if (mode == 9)
            printf("从 %c 移动到 %c，共 %d 层 ", st, ed, n);

        if (mode == 8 || mode == 9)
        {
            img_reset(n, st, ed, (speed ? last / 20 : 30));
            gotoxy(hout, 0, Y + 1);
            setcolor(hout, COLOR_BLACK, COLOR_WHITE);
        }

        if (mode == 4 || mode == 8 || mode == 9)
            put_pic();

        if (mode == 4 || mode == 8 || mode == 9)
        {
            printf("初始：");
            print_array();
        }
    }
    void work(const int mode, const int n, const char st, const char ed, const char tmp, const int speed, const int last)
    {
        reset(mode, n, st, ed, speed, last);
        move(mode, n, st, ed, n, st, ed, tmp, last);
    }

    //game part
    void move_operate(const int n, const int i, const int j)
    {
        if (!top[i])
        {
            printf("源柱为空!!!!");
            Sleep(750);
            return;
        }
        int top_i = stack[i][top[i]], top_j = stack[j][top[j]];
        if (top_i > top_j)
        {
            printf("大盘压小盘，非法移动");
            Sleep(750);
            return;
        }

        int tmp = pop(i);
        push(tmp,j);
        output(9, n, 0, 0, tmp, i + 'A', j + 'A', 1);
    }
    inline bool check(const int n, const char i)
    {
        if (top[i - 'A'] == n)
            return 1;
        return 0;
    }
    inline void clear()
    {
        const HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);
        gotoxy(hout, 0, 34);
        For(i, 1, width - 1)
            printf(" ");
```

```
        printf("\n");
        For(i, 1, width)
            printf(" ");
        gotoxy(hout, 0, 34);
    }
    void game_ver(const int n, const char st,
const char ed)
    {
        const      HANDLE      hout      =
GetStdHandle(STD_OUTPUT_HANDLE);
        char from, to;

        reset(9, n, st, ed, 0, 0);
        while (1)
        {
            if (check(n, ed))
            {
                gotoxy(hout, 0, 35);
                printf("游戏结束!!!!");
                return;
            }
            clear();
            printf("请输入移动的柱号(命令
形式:AC=A 顶端的盘子移动到 C) : ");
            if (scanf(" %c%c", &from, &to) !=
2)
                from = to = 'D';
            if (from >= 'a' && from <= 'z')
                from -= 32;
            if (to >= 'a' && to <= 'z')
                to -= 32;
            if  (from<'A'  ||  from>'C'  ||
to<'A' || to>'C' || from == to)
            {
                FFLUSH;
                continue;
            }
            FFLUSH;

            move_operate(n, from - 'A', to -
'A');
        }
    }

    //input part
    void pre_work(char *st, char *ed, char
*tmp)
    {
        bool f[3] = { 0 };
        if (*st >= 'a' && *st <= 'c')
            st -= 32;
        if (*ed >= 'a' && *ed <= 'c')
            ed -= 32;
        f[*st - 'A'] = 1;
        f[*ed - 'A'] = 1;
        if (!f[0])
            *tmp = 'A';
        else if (!f[1])
            *tmp = 'B';
        else
            *tmp = 'C';
    }
    void init_speed(int *speed, int *last)
    {
        while (1)
        {
            printf("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)：");
            if (!scanf(" %d", speed))
                *speed = 6;
            if (*speed < 0 || *speed>5)
            {
                FFLUSH;
                continue;
            }
            break;
        }

        if (*speed == 1)
            *last = 1000;
        else if (*speed == 2)
            *last = 500;
        else if (*speed == 3)
            *last = 200;
        else if (*speed == 4)
            *last = 50;
        else if (*speed == 5)
            *last = 1;
        else
            *last = 0;
    }
    void init_ed(const char st, char *ed)
    {
        while (1)
        {
            printf("请输入目标圆柱名
(A,B,C)：");
            if (!scanf(" %c", ed))
                *ed = 'D';
            if (!((*ed >= 'A' && *ed <= 'C')
|| (*ed >= 'a' && *ed <= 'c')))
            {
                FFLUSH;
                continue;
            }
            if (st == *ed)
            {
                printf("起始柱(%c)和目标柱
(%c)不能为同一个柱子\n", st, *ed);
                FFLUSH;
                continue;
            }
```

```
            return;
        }
    }
    void init_st(char *st)
    {
        while (1)
        {
            printf(" 请 输 入 起 始 圆 柱 名
 (A,B,C): ");
            if (!scanf(" %c", st))
                *st = 'D';
            if (!((*st >= 'A' && *st <= 'C')
|| (*st >= 'a' && *st <= 'c')))
            {
                FFLUSH;
                continue;
            }
            return;
        }
    }
    void init_n(int *n)
    {
        while (1)
        {
           printf("请输入盘子的个数(1-10):
");
            if (!scanf(" %d", n))
                *n = 11;
            if (n <= 0 || *n >= 11)
            {
                FFLUSH;
                continue;
            }
            return;
        }
    }
    bool init_mode(int *mode)
    {
        printf("[请选择 0-9] ");
        if (!scanf(" %d", mode))
        {
            FFLUSH;
            return 0;
        }
        if (*mode < 0 || *mode>9)
            return 0;
        return 1;
    }
    int init(int *mode, int *n, char *st, char
*ed, int *speed, int *last)
    {
        if (!init_mode(mode))
            return 0;
        if (!(*mode))
            return 1;
        FFLUSH;
        if (*mode != 5)
        {
            init_n(n);
            FFLUSH;
            init_st(st);
            FFLUSH;
            init_ed(*st, ed);
            FFLUSH;
        }
        if (*mode == 4 || *mode == 8)
        {
            init_speed(speed, last);
            FFLUSH;
        }

        return 2;
    }

    //screen operate
    void change_screen(const int mode)
    {
        if (mode == 1 || mode == 2)
            return;
        if (mode >= 3 && mode <= 7)
            system("mode    con    cols=100
lines=30");
        else if (mode == 8)
            system("mode    con    cols=100
lines=36");
        else
            system("mode    con    cols=100
lines=40");
    }
    void output_system()
    {
        For(i, 1, 35)
            printf("-");
        printf("\n");

        printf("1.基本解\n");
        printf("2.基本解<步数记录>\n");
        printf("3.内部数组显示<横向>\n");
        printf("4.内部数组显示<纵向+横
向>\n");
        printf("5.图形解-预备-画三个圆柱
\n");
        printf("6.图形解-预备-在起始柱子上
画n个盘子\n");
        printf("7.图形解-预备-第一次移动
\n");
        printf("8.图形解-自动移动版本\n");
        printf("9.图形解-游戏版\n");
        printf("0.退出\n");

        For(i, 1, 35)
```

```
            printf("-");
        printf("\n");
    }

    int main()
    {
        int mode, n, speed, last;
        char st, ed, tmp;

        while (1)
        {
            system("cls");
            system("mode    con    cols=80
lines=30");

            cnt = mode = n = speed = st = ed
= tmp = last = 0;

            output_system();

            int res = init(&mode, &n, &st,
&ed, &speed, &last);
            if (!res) // mode select wrong
                continue;
            else if (res == 1) // mode==0
                break;
            change_screen(mode);

            if (mode != 5)
                pre_work(&st, &ed, &tmp);

            if (mode == 1 || mode == 2 || mode
== 3 || mode == 4 || mode == 8)
                work(mode, n, st, ed, tmp,
speed, last);
            else if (mode == 5)
                put_pillar(30);
            else if (mode == 6)
                img_reset(n, st, ed, 30);
            else if (mode == 7)
            {
                img_reset(n, st, ed, 30);
                Sleep(500);
                if (n == 1)
                    move_disk(1, st, n, ed,
1, 30);
                else
                    move_disk(1,  st,  n,
tmp, 1, 30);
            }
            else //mode==9
                game_ver(n, st, ed);

            reset_xy(mode);
            printf("按回车继续");
            while (1)
            {
                char op;
                op = _getch();
                if (op=='\n' || op=='\r')
                    break;
            }
        }

        return 0;
    }
```