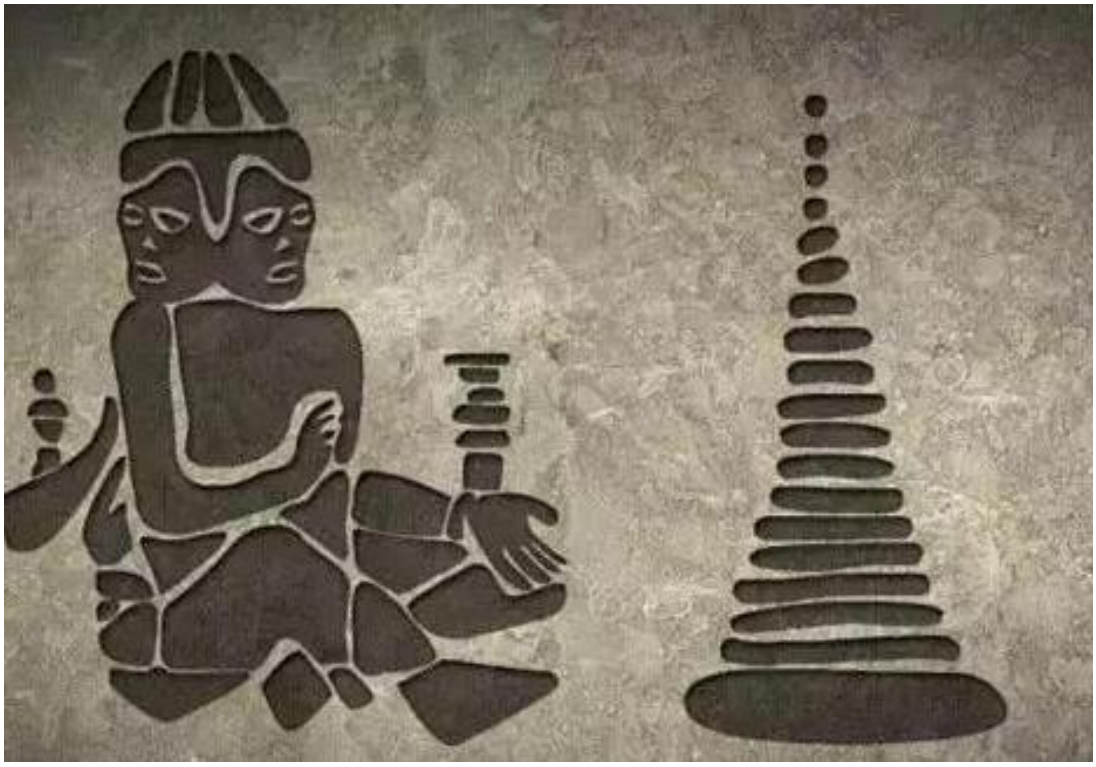
汉诺塔实验报告



班级 ：软件工程

作者姓名：陈君

学号 ： 2250420

日期 ： 12.4

汉诺塔实验报告

2250420 陈君

一、题目及基本要求描述

1.题目描述：

本次大作业要求将整合之前做的所有汉诺塔的各小题全部集成在一个程序中，制作一个菜单能够实现功能选择，并且加入cmd简单的图形化演示，实现图形解，具体要求完成汉诺塔综合演示的9个菜单项实现。

1.基本解：用递归的方法打印汉诺塔（Hanoi Tower）的正确移动步骤。汉诺塔的游戏规则：

三根杆：

游戏中有三根垂直的杆，通常标记为A、B和C。

圆盘：

有若干个圆盘，它们的直径不同，从小到大排列。开始时，所有圆盘都堆叠在一根杆上。

从小到大：

圆盘按直径递减的顺序从上到下堆叠，最大的圆盘在底部，最小的在顶部。

一次只能移动一个圆盘： 在任何时候，你只能移动一次一个圆盘，并且只能将它放在没有圆盘的杆上或者放在比它大的圆盘之上。

移动规则：

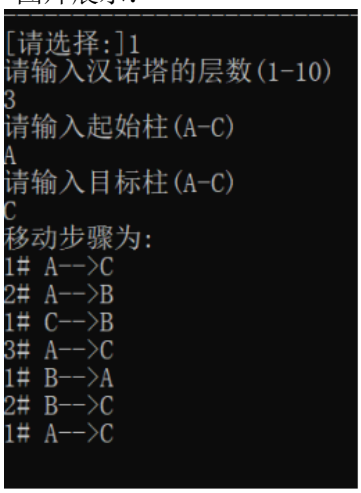
你可以将一个圆盘从一个杆上移动到另一个杆上，但必须遵循上面的规则。移动可以是从一个杆的顶部取下一个圆盘，然后放到另一个杆的顶部。

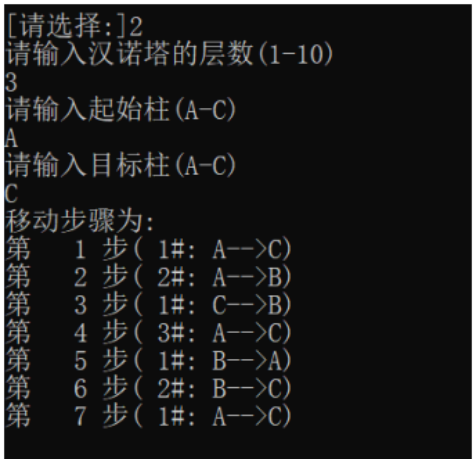
最终目标：

将所有的圆盘从起始杆移动到目标杆，可以借助中间的杆。

程序实现过程中，要求键盘输入汉诺塔的层数，起始柱（A-C）和目标柱（A-C）,同时要求能够实现错误处理，比如输入其他各种奇怪的字符，以及起始柱（A-C）和目标柱（A-C）不能相同。

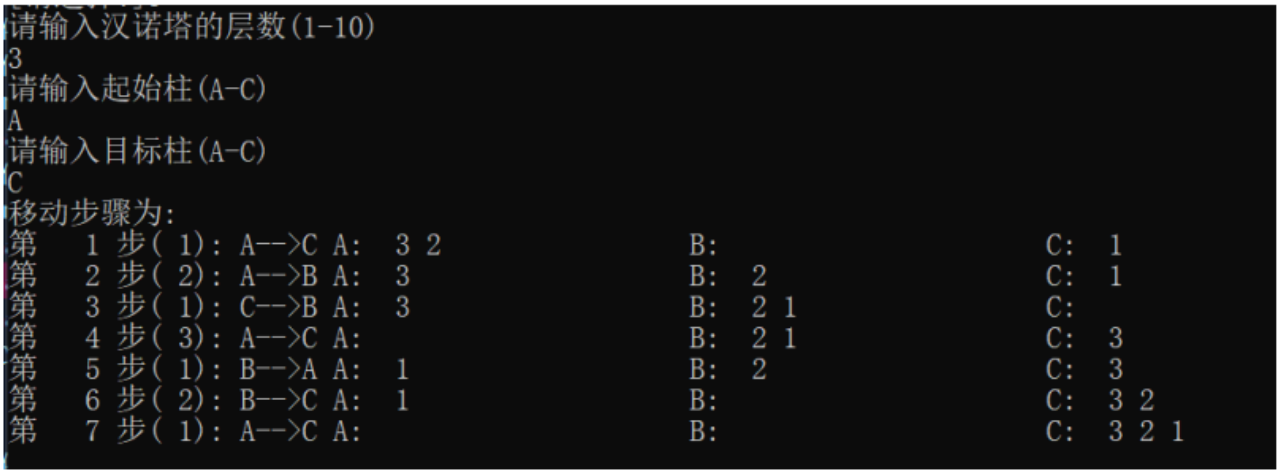
图片展示：



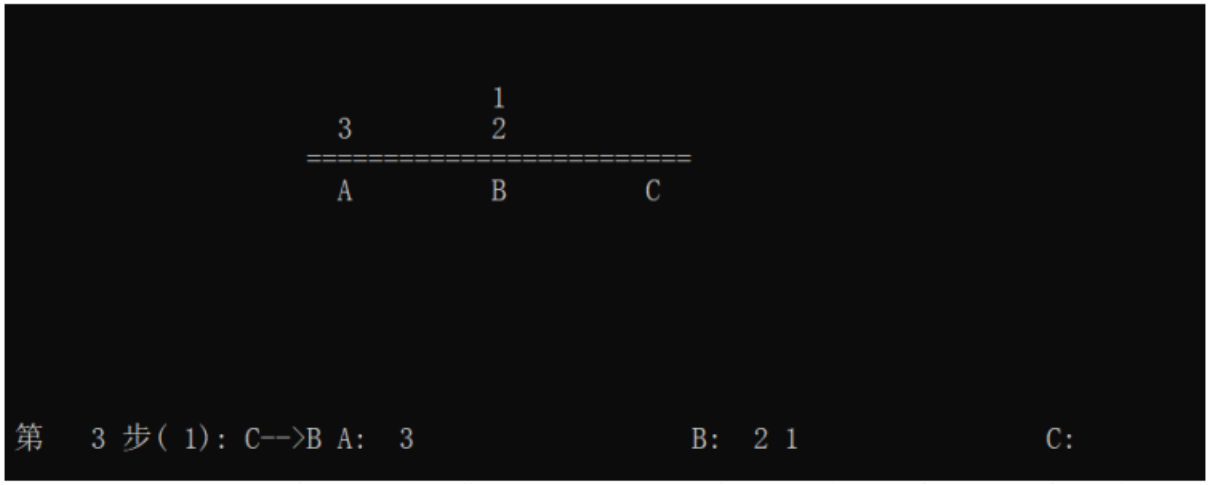


2. 基本解：除了第一题的要求外，增加了显示步数的额外要求。

3. 内部数组显示（横向）：除了基本解的要求外，需要给出每次移动时内部数组的情况，即在移动步数后额外显示三个存储柱子上盘子情况的数组。

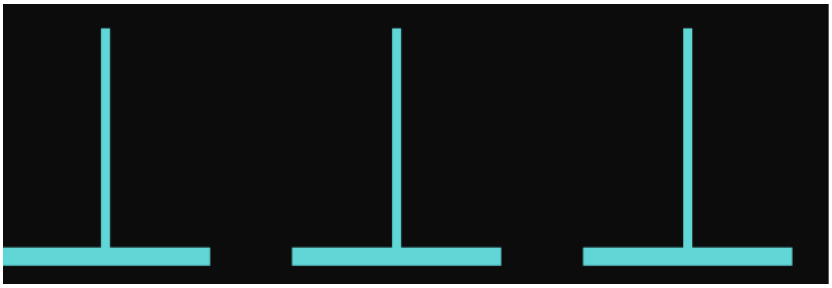


4. 内部数组显示（横向纵向）：除了第三题的要求外还要显示纵向的数组，即需要用字符画出汉诺塔的柱子，以及柱子上的盘子情况，这样可以让汉诺塔的移动更加直观。



5-9题是使用伪图形界面显示汉诺塔的移动，为了简化题目的难度，将显示汉诺塔的移动分解成了五个小题。

5. 图形解-预备-画三个圆柱：在屏幕上画出三根圆柱。要求加上延时函数，方便观察柱子产生的过



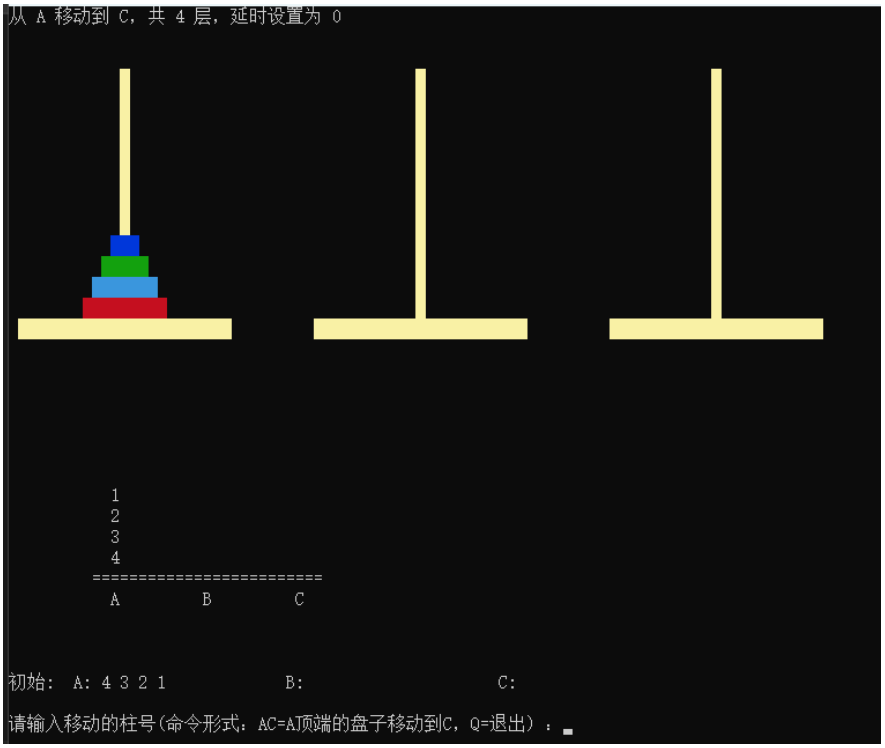
程

6. 图形解-预备在起始柱上画n个盘子：假设三根圆柱的编号从左到右分别为ABC，输入参数后，能画出汉诺塔圆盘的起始状态，要求圆盘颜色各不相同。要求加上延时函数，方便观察盘子产生的过程

7. 图形解-预备-第一次移动：在第六题的基础上，完成第一个圆盘的移动，需要圆盘移动顺序是先上移动，再平移，再下移。

8. 图形解-自动移动版本：调用5-7的选项，在移动盘子的时候能够显示出盘子的移动。

9. 图形解-游戏版：将整个界面做成游戏的形式，玩家可以输入移动的方向，然后能够实现盘子的移动，最后要求能够判断游戏结束的条件。



2. 基本要求：

为了更好的掌握函数的分解与应用技巧，对hanoi_nultiple_solutions.cpp中的函数做出一些限制，要求有：

1、只能使用一个递归函数

2、1/2/3/4/5/6/7/8的输入参数的函数要求共用

3、3/4/8中的共用一个函数横向输出后台的盘子情况

4、4/8中共用一个函数实现纵向输出

5、5/6/7/8/9中画柱子和盘子必须共用一个函数

6、7/8/9中盘子的移动必须共用一个函数

屏幕显示要求：

1、必要的时候需要添加延时函数，方便观察

2、每次只能用一次cct_cls()函数

全局变量要求：

只能设置四个全局变量，他们分别是：总移动步数，圆柱上现有圆盘的编号，圆柱上现有圆盘的数量，延时参数。

二、整体设计思路

主函数中调用showmenu（）提示用户输入指令，然后通过一个if-else的判断语句根据指令选择所需要执行的功能。

当指令为0时，直接执行exit（1）退出程序。

当指令小于等于4时，先提示用户输入参数，如果指令为4的话还要提示用户输入sleep的参数已经初始化汉诺塔的柱子和盘子，随后调用hanoi这个递归函数，递归地算出正确的移动步骤，在递归语句的移动过程中，会调用putoutbycommand函数，这个会根据command的值输出特定格式的移动步骤，具体实现会在下一个部分详细展示。

当指令为5-7时，也是先提示用户输入参数，然后调用menu_5_6_7(command,num,src,dst);函数，这个函数会根据指令的不同实现不同的功能。

当指令为8时，也是先提示用户输入参数，初始化可视化的柱子和盘子，接着调用hanoi（）递归的算出正确的盘子移动顺序。

当指令为9时，先提示用户输入参数，接着调用startgame函数，开始游戏。

三、主要功能的实现

void hanoi(int n, char src, char tmp, char dst, const int command);//递归调用函数

具体实现：如果只有一个盘子，直接将其从源柱子移动到目标柱子，输出移动步骤，可能包括图像的移动，根据命令选择执行额外的功能，比如func7中的图形化移动盘子。其他情况：将 n-1 个盘子从源柱子经过目标柱子移动到中间柱子，将第 n 个盘子从源柱子移动到目标柱子，输出移动步骤，可能包括图像的移动，根据命令选择执行额外的功能，比如func7中的图形化移动盘子。

void putout_by_command(const int command, int n, int step, char src, char dst);//输出函数

具体实现：根据command指令的不同，输出不同格式的移动步骤

Case1：标准输出格式，例如：1# A-->B

Case2：输出带步数的格式，第 1 步(1#: A-->B)

Case3：除了case2的输出外还要输出数组情况

Case4和8：除了case3的输出外还要输出纵向数组的情况

void move(char src, char dst);//移动盘子

具体实现：将plate数组的src上的顶上的数字交换到dst上的栈顶的位置，同时将num数组更新

void putout(int command);//输出当前数组情况

具体实现：使用for循环遍历plate数组然后输出

void initputout(char src, char dst, int n, int command);//输出化界面

具体实现：

当command=4时：调用输出盘子，输出柱子的函数，并且用cout输出一些提示信息，完成初始化

当command=8时：调用func5，func6，画出盘子和柱子，调用输出盘子输出柱子的函数，画出字符图

形的盘子和柱子，并且用cout输出一些提示信息，完成初始化。

void func5();//功能5

具体实现：使用cct_showch(column+i*32, row, ' ', color_y, color_x, 23);函数，按照demo打印特定的色块。

void func6(int n, char src);//功能6

具体实现：使用cct_showch(column+i*32, row, ' ', color_y, color_x, 23);函数，按照demo打印特定的色块。对于不同的盘子使用不同的颜色。

void func7(int plate_num, char src, char dst, int command);//功能7

具体实现：根据形参，算出盘子的目标位置，然后定义移动盘子的起始位置，使用while循环开始移动，先向上移动，实现移动的方式是，先打印盘子色块，Sleep()暂停一会，然后用黑色的色块覆盖原先盘子的色块，实现消失，然后再在上面一个的位置打印盘子色块，如此循环实现移动的效果。随后平移也是类似的方法，当盘子的x坐标和目标的x坐标相同时，结束，随后向下移动。

void startgame(char src,char dst,int num);//开始游戏

具体实现：先调用void initputout(char src, char dst, int n, int command);显示盘子，然后等待用户输入，会有错误输入处理，用户输入移动顺序后，调用func7函数移动盘子，同时更新后台的盘子数组，和num数组，最后会有判断游戏是否结束的条件：

目标柱子的盘子个数=最初要移动的总盘子数

游戏结束!。

四、调试过程碰到的问题

1.问题一

在做图形解的时候出现了成块成块的色块，而这些本来应该是原本显示文本的区域。

解决方法：经过仔细研读给出的cmd窗口的函数，发现使用了showch函数之后会将当前窗口的显示颜色替换为showch中设定颜色，此时需要再次调用setcolor函数，将颜色设置为默认值。

2.问题二

在做图形解的时候，盘子移动异常，和设定的位置有出入，移动位置靠下或者靠上。

解决方法：经过仔细的调试之后发现：是由于我的fuc7函数的逻辑和move函数的逻辑发生了冲突，在单独使用功能7时由于没有用move函数移动后台的记录盘子个数的数组，导致func7中参数出现了错误，只需要将func7加上条件如果command==7时加上一次move就可以了

3.问题三

在做图形解的时候，盘子移动过程中，柱子会消失。

解决方法：经过仔细的调试发现是由于清楚原先盘子的时候把柱子也删去了，再次显示盘子的时候没有把柱子复原，只要在showch盘子的时候，顺带复原柱子就行。

五、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：

对于汉诺塔问题，通过完成一系列相关题目，可以更好地理解和掌握递归算法的思想和应用。这个项目要求综合应用递归、用户输入处理、字符图形显示等多个方面的知识。在整合不同部分时，我深刻体会到如何将各个模块有机地结合起来，使得整个程序具有良好的结构。

在图形解的游戏版中，用户输入和交互是关键。我学到了如何设计一个简单而有效的用户界面，使用户能够方便地与程序进行交互。错误处理和用户提示也是至关重要的一部分，确保程序能够应对各种情况。

在处理字符图形显示时，我深入了解了如何使用终端库来在控制台上绘制简单的图形。掌握字符的排列和颜色设置，使得显示效果更加直观和美观。

在整个项目的编写过程中，我不断遇到各种问题，包括字符显示错误等。通过逐步构建、打印调试信息和逻辑分析，我提高了自己的调试技能，更好地理解和解决了问题。

这个项目包含多个任务，每个任务都有自己的难点和要求。我学到了如何有效地分解整体任务，逐步完成每个小任务，确保每个部分都能够正确工作。

5.2 在做一些较为复杂的程序时，分为若干小题优于直接一道大题的形式。

首先：将复杂任务分解为小题，使得每个小题都能独立完成，这有助于逐步推进项目，降低整体难度。其次，小题的分解有助于模块化设计，使得每个模块都专注于一个具体的功能或任务。这有助于提高代码的可维护性和可扩展性，比如当你发现我的程序中某个参数写错了，或者某个逻辑有问题，如果你直接一道大题，将所有的变量，逻辑耦合在一起，那么很有可能你需要将整个程序都要修改，这个过程将会十分痛苦，而将一个程序分解为小题，你可以通过这些模块的具体功能准确的定位到出错的位置，然后只要修改这一个地方就可以了。最后，小题可以更容易进行独立的测试，从而更容易发现和修复问题。这有助于确保每个功能都能正常工作。比如在这个汉诺塔问题中，每个功能都可以分解为一个小题，在测试时可以对每个功能进行独立测试，保证主代码的稳定性。

5.3 我在完成过程中确实考虑了前后小题的关联，而且也有效利用了前面小题的代码

比如在写func7的时候，我的显示盘子，柱子就是调用之前func5和6的函数，再如move函数也是多次使用，而且我的主要输出函数也是整合在了一起，这些都提要了代码的复用率，提高了我代码的简洁程度。

通过这次项目，我也积累了很多如何才能更好地重用代码的经验。

1、考虑关联性： 在设计每个小题时，要考虑模块之间的关联性。不同小题之间可能存在数据或逻辑的依赖关系。

2、定义清晰接口： 确保每个小题都有清晰的输入和输出接口，这样可以更容易地将它们连接在一起。

3、提取共用逻辑： 如果发现在不同的小题中存在相似的逻辑，考虑将这些逻辑提取为函数，以便在不同的上下文中重复使用。

4、参数化： 将函数设计成可接受参数，以适应不同的需求。例如，在处理不同汉诺塔层数时，可以将层数作为参数传递给函数。

5、构建工具函数： 如果有一些通用的功能，例如用户输入处理、错误处理、字符显示等，可以将这些功能构建成工具函数或者放入一个公共库中，方便在不同小题中重用。

5.4 在本次作业中，可以通过以下方式更好地利用函数来编写复杂的程序：

1. 模块化设计： 将程序划分为小模块，每个模块执行特定的任务。每个模块可以由一个或多个函数组成，这样可以使代码更加清晰，每个函数负责一个特定的功能。比如在最后实现功能8的过程中，将这个功能分解为了5，6，7三个功能，分别显示盘子，柱子，实现盘子的移动，这样子可以让实现功能8的时候思路更加清晰，同时也使得功能8这个复杂功能的代码实现更加简洁，极大的提高了代码的可读性，提高了完成这个程序的效率，而且分解为小的函数可以在调试的时候很方便的定位到到底是哪个模块出现了错误。

2. 参数化设计： 设计函数时，考虑将函数参数化，以便在不同的场景中重复使用。通过灵活使用参数，可以使函数更通用、适用范围更广。比如在我的程序中很多函数都传入了command这个参数，因为很多选项中所需要实现的函数共用了很多代码，传入command参数就可以通过条件判断实现代码的微调，减少了代码量，提高了写代码的效率

3.通过函数的返回值来实现错误处理：比如在我的程序中，在游戏版功能中需要判断用户输入是否正确，我使用了is_valid函数判断是否正确最后返回一个bool类型的值，返回判断的情况，时主函数startgame更加简洁。

附件：源程序

主函数：

```
/* 软工 250420 陈君 */
#include<iostream>
#include"cmd_console_tools.h"
#include"hanoi.h"
int main()
{
 int command,num;//声明选项和汉诺塔层数
 char src, dst,temp;//声明柱子
 /* demo中首先执行此句，将cmd窗口设置为40行x120列（缓冲区宽度120列，行数9000行，即cmd窗口右侧带有垂直滚动杆）*/
 cct_setconsoleborder(120, 40, 120, 9000);
 while (true)
 {
     command = showmenu();//调用menu中的展示UI
```

```
        if (command <= 4)
        {
            if (command == 0)
                exit(1);
            setparameter(&src, &dst, &temp, &num);//初始化参数
            if (command == 4)
            {
                setsleep();
                initputout(src, dst, num, command);//功能4的初始化
            }
            hanoi(num, src, temp, dst, command);
            if (command < 4)
                std::cout << "按回车键继续";
            clear_screen();
        }
        else if (command >= 5&&command <=7)
        {
            setparameter(&src, &dst, &temp, &num);//初始化参数
            menu_5_6_7(command,num,src,dst);
            to_be_continued(NULL);
        }
        else if (command == 8)
        {
            setparameter(&src, &dst, &temp, &num);//初始化参数
            setsleep();
            initputout(src, dst, num, command);//初始化
            hanoi(num, src, temp, dst, command);
            to_be_continued(NULL,0,35);
        }
        else if (command == 9)
        {
            setparameter(&src, &dst, &temp, &num);//初始化参数
            initputout(src, dst, num, command);//初始化
            startgame(src, dst, num);//开始游戏
            to_be_continued(NULL, 0, 38);
        }
    }
    return 0;
}
```

重要实现函数：

```
int plate[3][10];
int num[3];//记录柱子上的盘子数
int step = 0;
int sleep;
void hanoi(int n, char src, char tmp, char dst,const int command)
```

```
{
 if (n == 1) //A   B   C
 {
      step++;
      putout_by_command(command, n, step, src, dst);//后台数组移动完之后图像开始移动
      if(command==8)
          func7(n, src, dst,command);
      return;
 }
 hanoi(n - 1, src, dst, tmp,command);
 step++;
 putout_by_command(command, n, step, src, dst);
 if(command==8)
      func7(n, src, dst,command);
 hanoi(n - 1, tmp, src, dst,command);
}
void putout_by_command(const int command, int n, int step, char src, char dst)
{
 switch (command)
 {
 case 1:
      if(n==1)
      cout << " 1# " << src << "-->" << dst << endl;
      else
      cout << setiosflags(ios::right) << setw(2) << n << "# " << src << "-->" << dst << endl;
      break;
 case 2:
      if (n == 1)
           cout << "第" << setiosflags(ios::right) << setw(4) << step << " 步" << "(" <<
           setiosflags(ios::right) << setw(2) << "1" << "#: " << src << "-->" << dst<<")" << endl;
      else
           cout << "第" << setiosflags(ios::right) << setw(4) << step << " 步" << "(" <<
           setiosflags(ios::right) << setw(2) << n << "#: " << src << "-->" << dst << ")" <<endl;
      break;
 case 3:
      if (n == 1)
           cout << "第" << setiosflags(ios::right) << setw(4) << step << " 步" << "(" <<
                setiosflags(ios::right) << setw(2) << "1" << "): " << src << "-->" << dst;
      else
           cout << "第" << setiosflags(ios::right) << setw(4) << step << " 步" << "(" <<
                setiosflags(ios::right) << setw(2) << n << "): " << src << "-->" << dst;
      move(src, dst);
      putout(command);
      break;
 case 4:
```

```
        sleeptime(sleep);
        cct_gotoxy(0, 17);
        if (n == 1) //A   B   C
            cout << "第" << setiosflags(ios::right) << setw(4) << step << "步" << "(" <<
                "1" << "#: " << src << "-->" << dst<<") ";
        else
            cout << "第" << setiosflags(ios::right) << setw(4) << step << "步" << "(" <<
                n << "#: " << src << "-->" << dst << ") ";
        move(src, dst);
        putout(command);
        //sleeptime(sleep);
        putoutplate(command);
        break;
    case 8:
        sleeptime(sleep);
        cct_gotoxy(0, 32);
        cct_setcolor();
        if (n == 1) //A   B   C
            cout << "第" << setiosflags(ios::right) << setw(4) << step << "步" << "(" <<
            "1" << "#: " << src << "-->" << dst << ") ";
        else
            cout << "第" << setiosflags(ios::right) << setw(4) << step << "步" << "(" <<
            n << "#: " << src << "-->" << dst << ") ";
        move(src, dst);
        putout(command);
        //sleeptime(sleep);
        putoutplate(command);
        break;
    }
}
/*****************************************************************************
  函数名称：move
  功    能：移动盘子
  输入参数：const int command, int n, int step, char src, char dst
  返 回 值：空
  说    明：
*****************************************************************************/
void move(char src, char dst)
{
 plate[dst - 'A'][num[dst - 'A']] = plate[src - 'A'][num[src - 'A'] - 1];
 plate[src - 'A'][num[src - 'A'] - 1] = 0;
 num[dst - 'A']++;
 num[src - 'A']--;
}
void func5()//功能5
```

```
{
 int color_x = 14;
 int color_y = 14;
 int row = 15;
 int column = 1;
 for (int i = 0; i < 3; i++)
        cct_showch(column+i*32, row, ' ', color_y, color_x, 23);
 for (int i = 0; i < 12; i++)
 {
     for (int j = 0; j < 3; j++)
     {
          cct_showch(column+11+j*32, row-i-1, ' ', color_y, color_x, 1);
          Sleep(100);
     }
 }
}
/****************************************************************************
  函数名称：func6
  功    能：功能6,显示盘子
  输入参数：int n,char src
  返 回 值：空
  说    明：
****************************************************************************/
void func6(int n,char src)
{
 int row = 15;
 int column = 1;
 func5();//先显示柱子
 for (int i = 0; i < n; i++)
 {
     cct_showch(12+32*(src-'A')-n+i, row - 1 - i, ' ', n - i, n - i, (n - i) * 2 + 1);
     Sleep(100);
 }
}
/****************************************************************************
  函数名称：func7
  功    能：功能7,移动盘子
  输入参数：int n,char src，char dst
  返 回 值：空
  说    明：
****************************************************************************/
void func7(int plate_num, char src, char dst,int command)   //功能7
{
 if(command==7)
 move(src, dst);
```

```
int src_hight = num[src - 'A']+1;
int dst_hight = num[dst - 'A']-1;
int row = 15;
int column = 1;
int sleeptime = 100;
//初始化盘子位置
int x = 12+32*(src-'A')-plate_num;//移动盘子的x坐标
int y = row-src_hight;//移动盘子的y坐标
int dst_x = 12 + 32 * (dst - 'A') - plate_num;
int dst_y = row - dst_hight-1;
if (command == 7)
{
    while (_getch() != '\r')
        ;

}
//向上运动出柱子
while (y >1)
{
    cct_showch(x, y, ' ', plate_num, plate_num, plate_num * 2 + 1);
    Sleep(sleeptime);
    cct_showch(x, y, ' ', 0, 0, plate_num * 2 + 1);
    if(y>=3)
    cct_showch(x+plate_num, y, ' ', 14, 14, 1);
    Sleep(sleeptime);
    y--;
}
//横向移动
while (x != dst_x)
{
    cct_showch(x, y, ' ', plate_num, plate_num, plate_num * 2 + 1);
    Sleep(sleeptime);
    cct_showch(x, y, ' ', 0, 0, plate_num * 2 + 1);
    Sleep(sleeptime);
    x = x + ((src > dst) ? -1 : 1);
}
while (y < dst_y)
{
    cct_showch(x, y, ' ', plate_num, plate_num, plate_num * 2 + 1);
    Sleep(sleeptime);
    cct_showch(x, y, ' ', 0, 0, plate_num * 2 + 1);
    if (y >= 3)
        cct_showch(x + plate_num, y, ' ', 14, 14, 1);
    Sleep(sleeptime);
    y++;
```

```
 }
 cct_showch(x, y, ' ', plate_num, plate_num, plate_num * 2 + 1);
 cct_setcolor(); //恢复缺省颜色
}
/****************************************************************************
  函数名称：menu_5_6_7
  功    能：5，6，7的菜单调用函数
  输入参数：int command, int num, char src, char dst
  返 回 值：空
  说    明：
****************************************************************************/
void menu_5_6_7(int command, int num, char src, char dst)
{
 cct_cls();
 switch (command)
 {
 case 5:
      func5();
      break;
 case 6:
      func6(num,src);
      break;
 case 7:
      cout << "从 A 移动到 C，共 4 层";
      func6(num, src);
      func7(1,src, 'A' + 'B' + 'C' - src - dst,command);
      break;
 }
}
void startgame(char src,char dst,int n)//开始游戏
{
 cct_setcolor();//设置颜色
 while (1)
 {
      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 now_src, now_dst;
      while (1)
      {
           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) ：";
           now_src = _getch();
           now_dst = _getch();
           if (now_src >= 'a')
                now_src = now_src - 32;
```

```
            if (now_dst >= 'a')
                now_dst = now_dst - 32;
            if (now_src >= 'A' && now_src <= 'C' && now_dst >= 'A' && now_dst <= 'C' && now_src != now_dst,is_valid(now_src,now_dst))
                break;
        }
        step++;
        putout_by_command(8, plate[now_src - 'A'][num[now_src - 'A'] - 1], step, now_src, now_dst);//后台数组移动完之后图像开始移动
        func7(plate[now_dst-'A'][num[ now_dst - 'A']-1], now_src, now_dst, 8);
        if (plate[now_dst - 'A'][num[now_dst - 'A'] - 1] == 1 && num[now_dst - 'A'] == n)
        {
            cct_gotoxy(0, 35);
            cout << "游戏结束!";
            break;
        }
    }
   }
```