**汉诺塔的综合演示**

**班级：信息类11班**

**学号：2151294**

**姓名：马威**

**完成日期：2021.12.9**

1. **题目：集成所有之前做过的有关汉诺塔的小题，并加入图形化演示，用菜单项进行选择**
   1. **基本解**

**显示每一步的移动情况，包括盘子的编号以及每一步起始柱、目标柱的编号**

**1.2.基本解（步数记录）**

**显示每一步的移动情况，以及每一步的步数**

**1.3.内部数组显示（横向）**

**显示每一步的移动情况，并以一横排的形式输出每根柱子上盘子编号、数量的实时情况**

**1.4.内部数组显示（纵向+横向）**

**在1.3的基础上，加上内部数组纵向输出，即以数字堆叠的形式模仿三根柱子上的盘子。此时每一步移动需擦除原有的内容后再输出这一步的内容。**

**1.5.图形解-预备-画三个圆柱**

**使用伪图形界面工具，画出汉诺塔的三根柱子。为了方便观察实现过程，需要加延时。**

**1.6.图形解-预备-在起始柱上画n个盘子**

**要求输入起始圆柱的编号，盘子的数量，在指定的起始圆柱上从小到大画n个盘子，其颜色各不相同。为了方便观察实现过程，需要加延时。**

**1.7.图形解-预备-第一次移动**

**1.在1.6的基础上，显示第一步移动的动画。**

**2.第一步不一定是源柱->目标柱，也可能是源柱->中间柱，所以一定要调用递归函数，指导第一步的行动。**

**3.不允许直接在两个圆柱间移动，必须先上移、再平移、再下移。**

**1.8.图形解-自动移动版本**

**在1.7的基础上，完整地显示每一步移动的动画，同时也进行内部数组的横向与纵向输出。每次圆盘的移动方式也必须是上移、平移、下移。**

**1.9.图形解-游戏版**

**1.每次键盘输入两个字母（A-C）之间，大小写均可，表示本次移动的源柱和目标柱。**

**2.移动时要检查合理性，若不符合移动规则（大盘压小盘、源柱为空等）要提示出错并重输，每次合理的移动都必须记录步数。**

**3.每次圆盘的移动方式也必须是上移、平移、下移。**

**4.待所有盘子按序移动到结束柱则提示“游戏结束”**

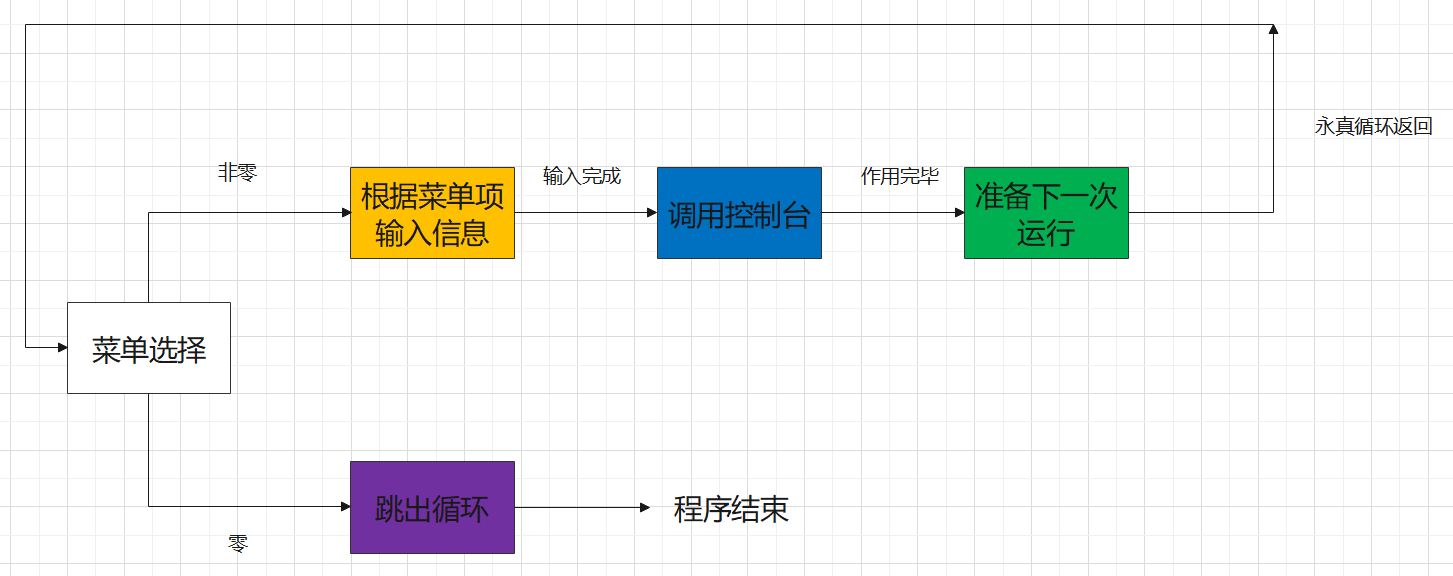
**1.10.退出**

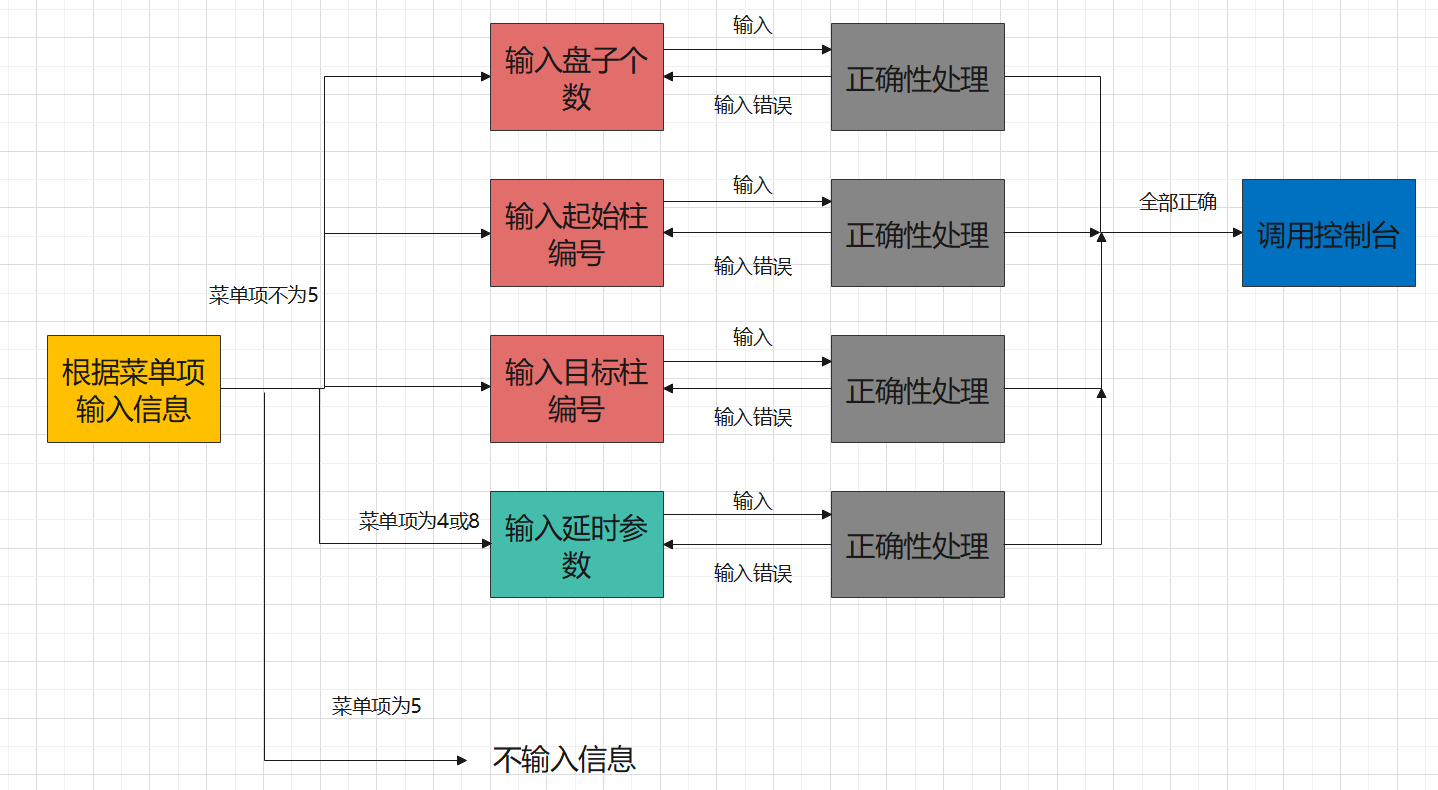
**按下数字0能直接退出程序。**

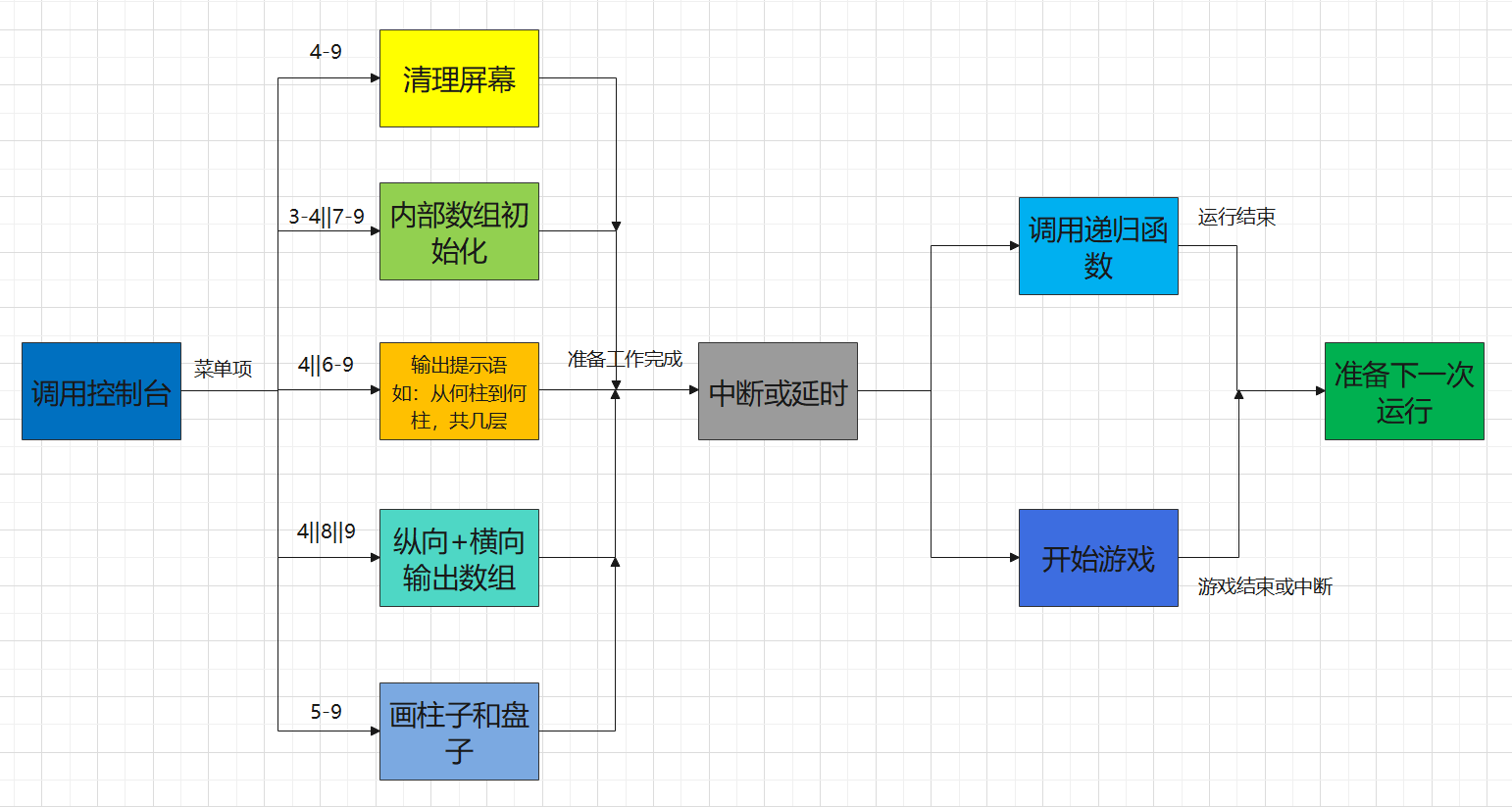
1. **整体设计思路**
2. **整个程序以一个永真循环为背景，通过菜单函数来控制菜单选择。**
3. **若菜单项为零，直接跳出循环，程序结束。**
4. **若菜单项不为零，调用一个总的输入函数（里面包含了多种正确性处理的函数），输入信息后再调用一个总控函数（里面根据菜单项选择调用不同函数，作用相当于一个控制台）实现主要功能。**
5. **递归函数只有一个，所以第3点调用的是调用递归函数前，准备工作中需要的函数。而在递归过程中需要同时实现很多功能，所以在递归函数中会调用一个伴随函数，其通过菜单函数的返回值选择调用不同函数。**
6. **根据不同的主要功能，有针对性地写出完成各种功能的函数。**
7. **主要功能的实现**

**3.1.主函数及其有关函数**

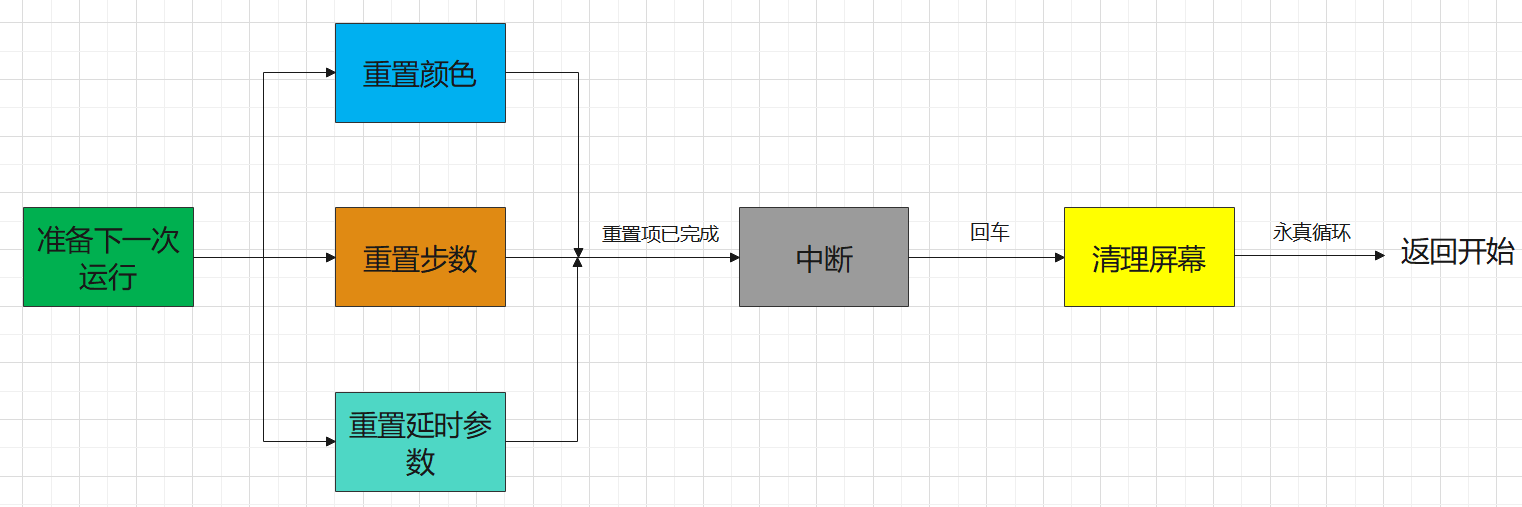
**1、主函数的结构**

****

**2、输入函数的结构**

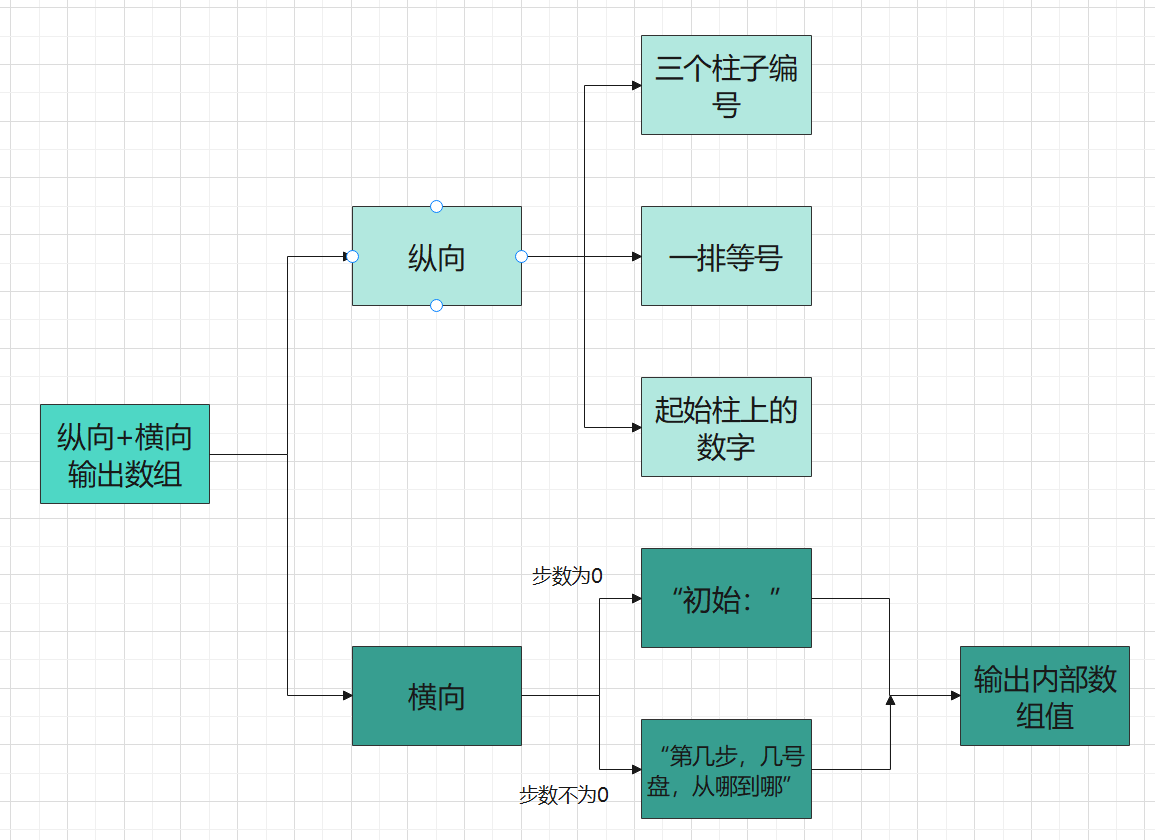
1. **控制台函数的结构**

**4、重设函数的结构**

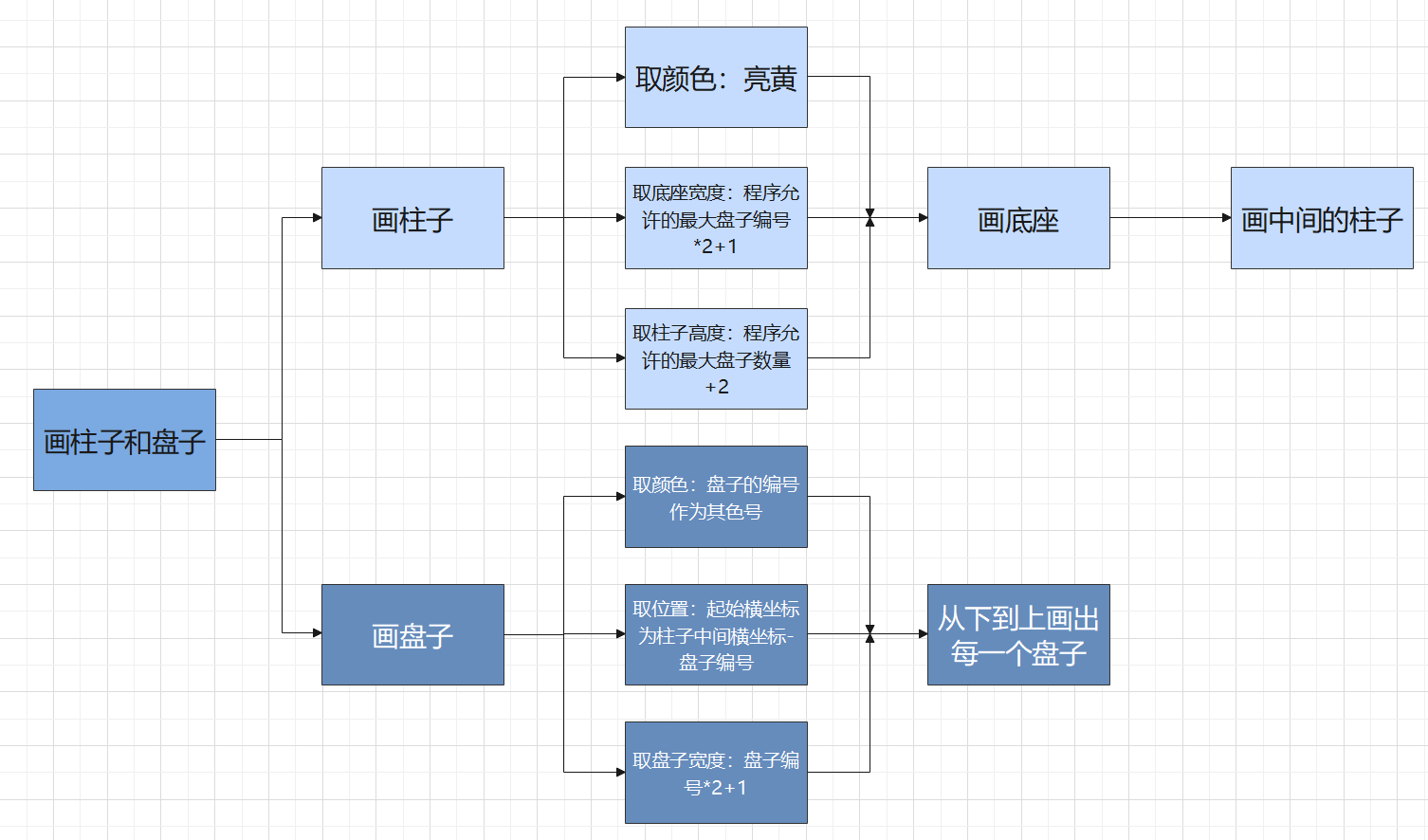
****

**3.2.控制台有关函数**

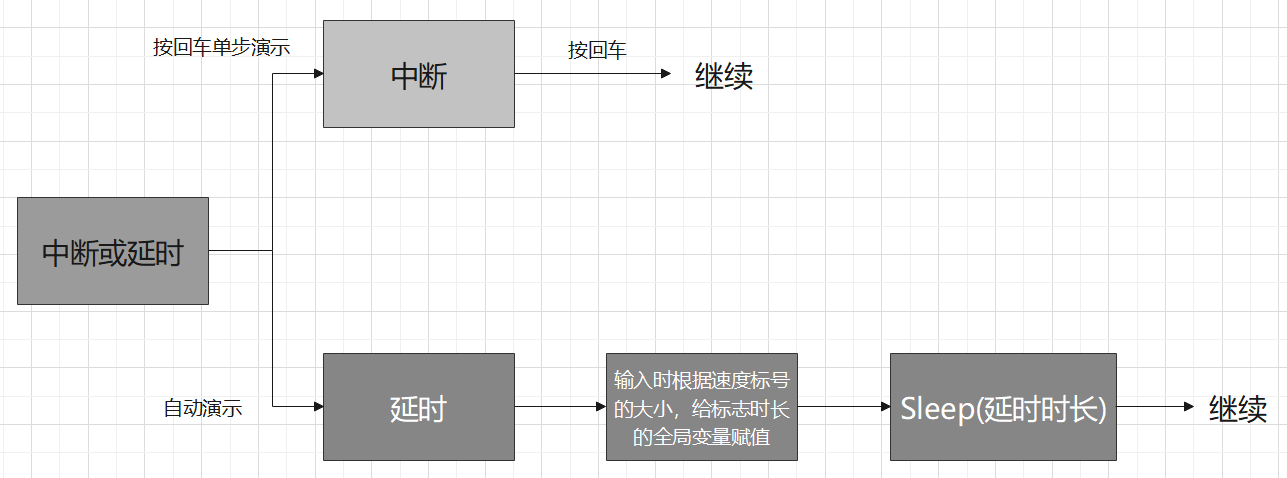
**1、纵向、横向输出数组**

****

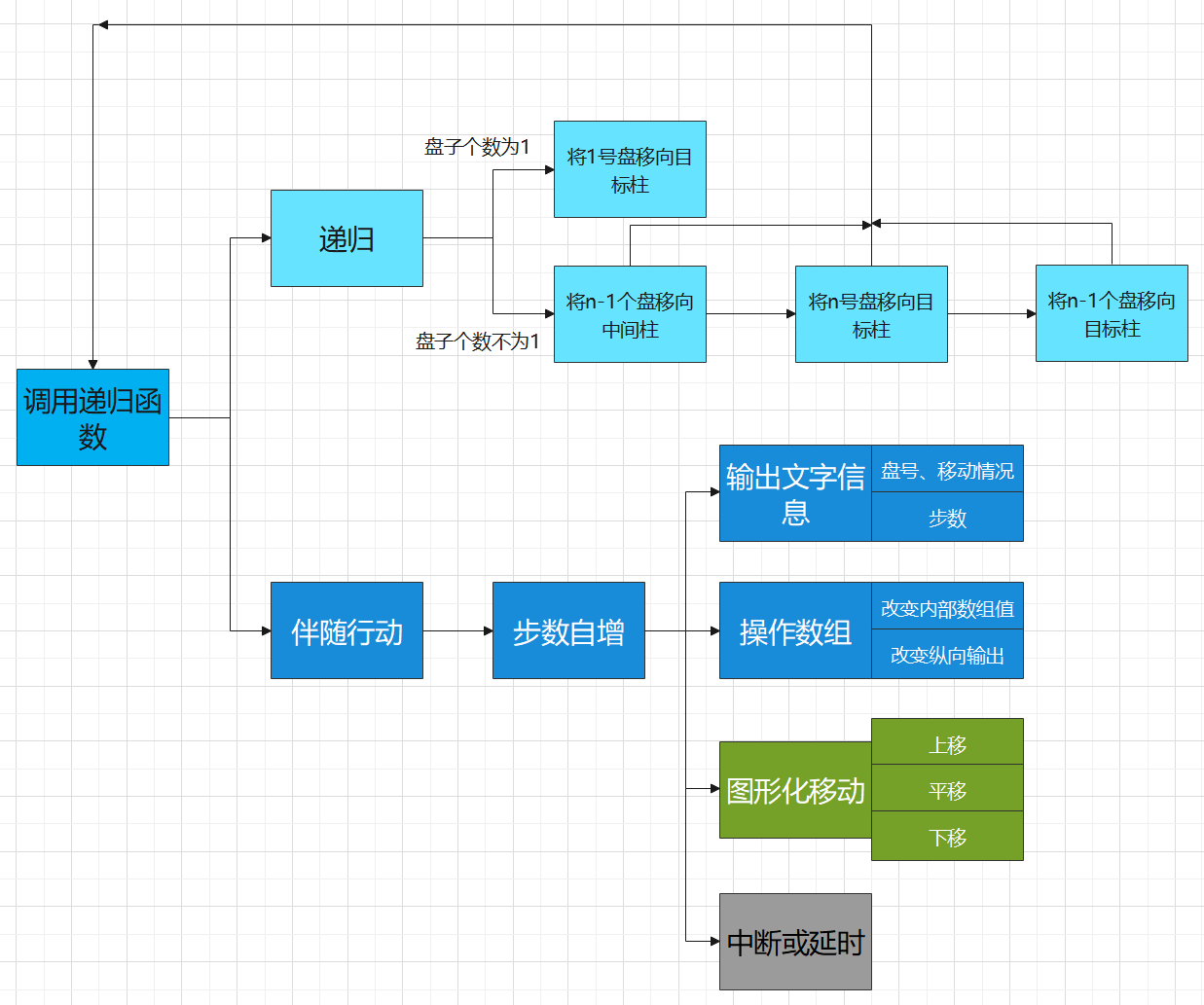
**2、画柱子、画盘子**

****

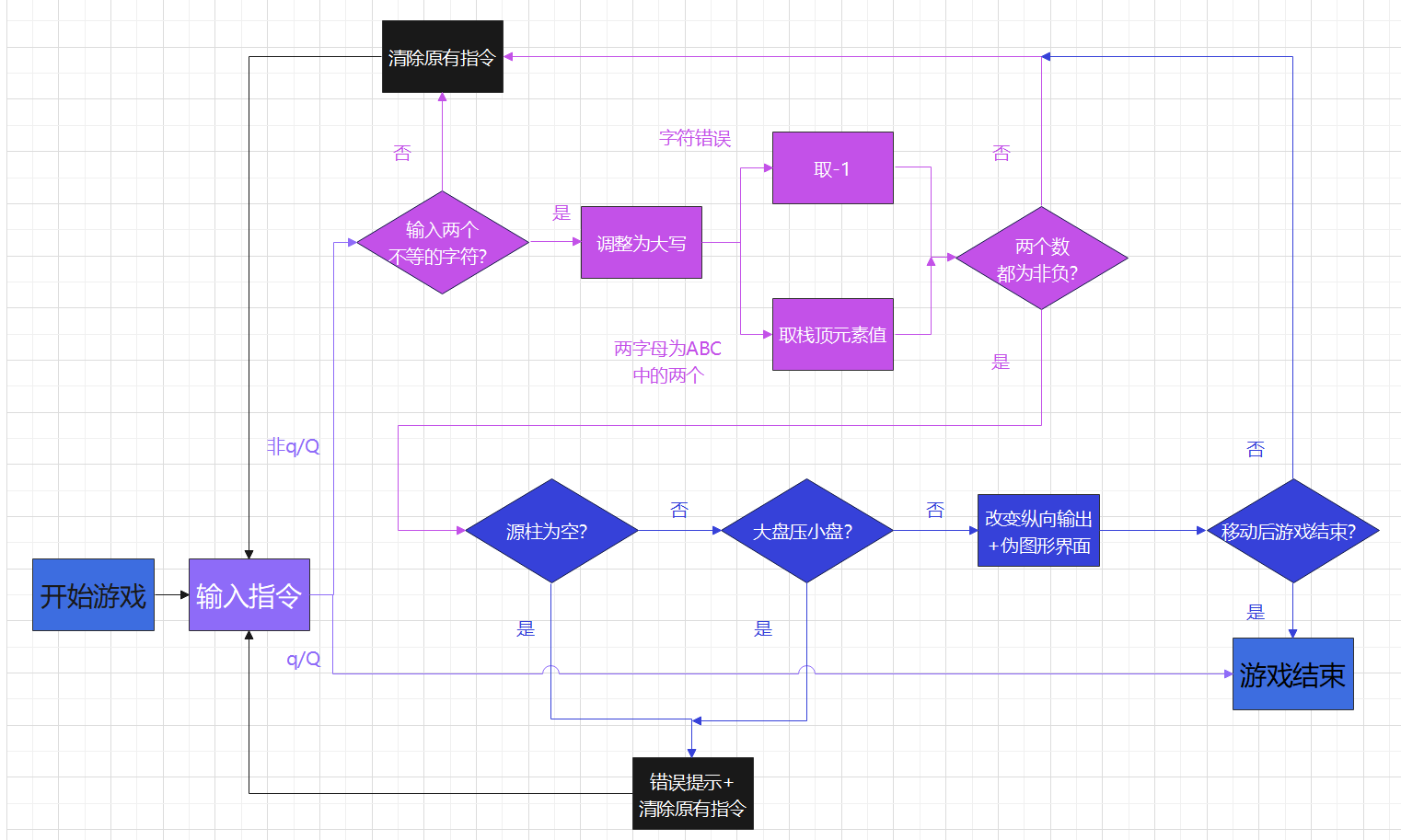
**3、间断点**

****

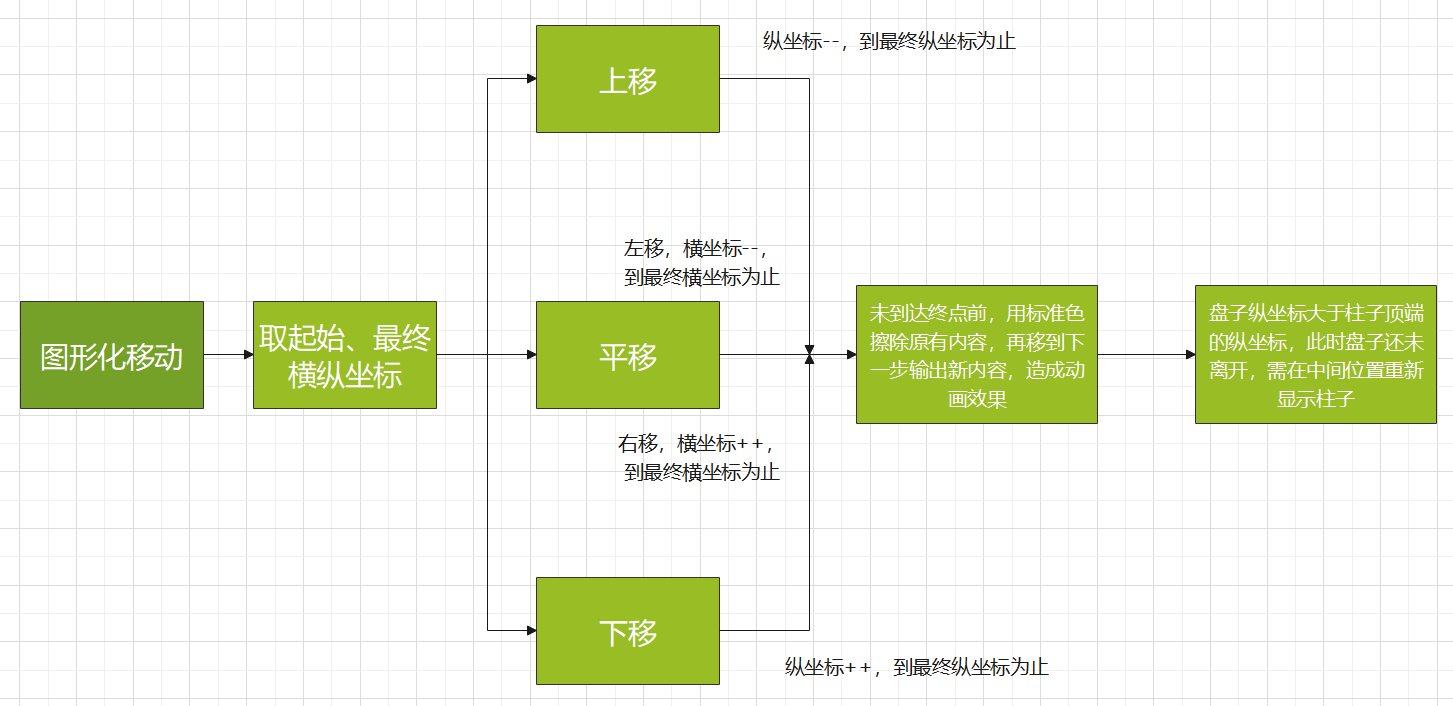
**4、递归函数**

****

**5、游戏**

****

**6、图形化移动**

****

1. **调试过程碰到的问题**

**问题：写菜单项8的时候，横向、纵向数组显示正确，但伪图形界面里盘子移动总是不正确。**

**原因：查看递归函数中调用的伴随函数，发现逻辑上的错误：横向、纵向数组改变后，内部数组值也改变了，但盘子移动的函数需要利用移动前的内部数组值，所以产生了错误。**

**解决：只要在调用显示盘子移动的函数前，再调用一次改变内部数组的函数，撤销这一步移动带来的数组值改变，让内部数组值回到移动前的状态，伪图形界面里盘子移动就正确了。**

**5. 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**

**1.解决这类比较复杂且规模不小的问题时，先从整体入手，将整体设计思路写下来，着手将大框架的程序尝试写出来，比如规划主函数的大致内容、菜单函数的形式等。**

**2.在大框架下，按一定顺序（本题中按照菜单项的顺序）将小单元写出，先思考需要哪些功能，函数如何划分；再看看做过的小题中是否有可以使用的代码。总的来说就是逐个击破各个小问题，最终解决大问题。**

**3.在写具体函数的实现时，不要想到什么就一头扎进去写，要瞻前顾后。**

**4.瞻前：看看已经写出的代码，要写下来的是否与已有的重复？若重复了，如何在不影响功能的前提下用更精简而有条理地代码完成？然后再去修改、调试。**

**5.顾后：把自己想象成这个程序的客户，仔细思考自己对于这个产品是否会有不满的地方，会提出什么更高的要求，再把思路转换回来，想想面对这样的问题，是否只要在写程序时将方法稍微改变一下，后期维护升级时就会更轻松。**

**6.做第5点不宜太晚，应尽早规划好哪些地方在这个题目中可能是要“写死”的，但实际上升级时需要它是“活着”的，否则回想起来时，自己已经“写死”了太多的量，还想再改进时，工作会变得非常困难。**

**7.针对5、6点，一定要能用、会用、善用全局const和#define，不要在写具体实现时嫌写一个变量名比一个“死板的”数字来的麻烦，用上这些，日后只要通过改变参数的值就可以达到“牵一发而动全身”的改进效果。**

**5.2.在做一些较为复杂的程序时，是分为若干小题还是直接一道大题的形式更好？**

**分为若干小题更好。**

**从着手难度看：直接一道大题很难看一眼就直接上手写代码，至少需要构思总体的设计思路，再从比较简单的一些问题的解决上出发，而这已经需要不少的时间了。相比之下，分为若干个小题，在做每一个小提时，思考的问题不会太复杂，很快就能找到问题的突破口，仅仅对于完成题目而言，效率的确会快一点。**

**从代码利用看：直接做一道大题，全部的内容都是新的，需要“白手起家”，工作量不小。而分为多个小题后，较为复杂的小题总会使用到比较简单的小题中实现的功能，这时候就给了自己一次重新检视、利用、改进自己原先代码的机会，相当于再检查一次自己写的东西，总可以看出可以改进提升的地方；同时工作量也会小一些。**

**5.3.汉诺塔完成了若干小题，总结你在完成过程中是否考虑了前后小题的关联关系，是否能尽可能做到后面小题有效利用前面小题已完成的代码，如何才能更好地重用代码？**

**在完成过程中考虑到了前后小题之间的关联，也尽可能做到了利用已完成的代码。**

**重用代码不是简单的复制粘贴，能用就行。如5.2中所言，重用已有代码是给自己一个重新检视自己代码的机会，总的来说要重新检查三个方面：**

**1、能利用的地方有多少？不必利用、不能利用的部分是哪里？**

**2、是否精简、逻辑清晰明了？若不是，冗余代码如何寻找，又如何在不影响功能的前提下改进代码呢？**

**3、是否存在隐藏的逻辑错误？是否存在违反之前题目规定的做法？这些做法现在能用吗？这些做法会对自己和程序有什么影响？**

**总之，更好地重用代码，不能无所作为，而要努力思考、改错、改进、提升，然后才能更好地利用，否则有些问题可能越积越多，终会造成严重的后果。**

**5.4.以本次作业为例，说明如何才能更好地利用函数来编写复杂的程序**

**在构思整体设计思路的时候，会划分程序的各个主要功能。使用函数实现这些功能，在程序里一个复杂功能的实现就可以浓缩成一句函数的调用，使得逻辑非常的清晰明了，也能不难看出整个程序的结构。所以合理划分好各项需要实现的功能，再逐一使用函数实现，对于编写复杂程序会大有帮助。**

**例1：菜单项7、8中显示盘子的移动动画。**

**这个功能不是一个单一功能，它包括上移、平移、下移等多种功能，将其本身作为一个总的函数，其中调用上移函数、平移函数、下移函数等实现单一功能的函数，这样就很有条理地完成这个功能了。**

**例2：菜单项9的游戏**

**这一项包含的功能更多，包括输入、判断、赋值、移动等。可以先以菜单项9整体作为一个总的函数，找出高频使用的功能，每一个都用函数来实现，按需取用，如：**

**①消除上次输入的指令**

**②根据输入的指令给起始柱标号等变量赋值**

**③用②结果判断是否能移动**

**④输出移动错误的信息并擦除**

**⑤根据②的结果，调用多个函数完成移动**

**......**

**这样菜单项9的程序结构就比较清晰了，就算其功能比较复杂，完成起来还是比较有条理的。**

**6. 附件：源程序**

**1.hanoi\_multiple\_solutions.cpp**

/\*2151294 信11 马威\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<iostream>

#include<iomanip>

#include<conio.h>

#include<Windows.h>

#include"cmd\_console\_tools.h"

#include"hanoi.h"

using namespace std;

static int \_\_stepnumber = 0;

static int \_\_delay\_time;

int TOWER[3][PLATE\_NUM\_LIMIT];

int MARK[3];

void hanoi(int number\_of\_plates, char source\_tower, char temporary\_tower, char destination\_tower, const int menu)

{

if (number\_of\_plates == 1) {/\*若汉诺塔层数为1，将该盘从起始柱直接移向目标柱\*/

\_\_stepnumber++;

hanoi\_act(number\_of\_plates, source\_tower, destination\_tower, menu);

}

else { /\*若汉诺塔层数不为1\*/

hanoi(number\_of\_plates - 1, source\_tower, destination\_tower, temporary\_tower, menu); /\*将n-1个盘子移向中间柱\*/

\_\_stepnumber++; /\*步数自增\*/

hanoi\_act(number\_of\_plates, source\_tower, destination\_tower, menu); /\*将第n个盘子直接移向目标柱\*/

hanoi(number\_of\_plates - 1, temporary\_tower, source\_tower, destination\_tower, menu); /\*将其余n-1个盘子移向目标柱\*/

}

}

void hanoi\_act(int number\_of\_plates, char source\_tower, char destination\_tower, const int menu)

{

hanoi\_act\_info(number\_of\_plates, source\_tower, destination\_tower, menu); /\*1、输出每一步的文字信息\*/

hanoi\_act\_array(number\_of\_plates, source\_tower, destination\_tower, menu); /\*2、对数组进行操作\*/

/\*3、菜单项7或8：图形输出+再次改变数组值\*/

/\*注：菜单项7只输出第一步的移动\*/

if ((menu == 7 && \_\_stepnumber == 1) || menu == 8) {

console\_show\_move(number\_of\_plates, source\_tower, destination\_tower, menu);

array\_move\_plates(source\_tower, destination\_tower);

}

/\*4、菜单项4或8：每一步后的暂停或延时\*/

if (menu == 4 || menu == 8) {

pause(\_\_delay\_time); /\*每一步后的暂停或延时\*/

}

}

void hanoi\_act\_info(int number\_of\_plates, char source\_tower, char destination\_tower, const int menu)

{

/\*菜单项1：打印每一步的解\*/

if (menu == 1)

cout << number\_of\_plates << "# " << source\_tower << "---->" << destination\_tower << endl;

/\*菜单项2或3：打印每一步的步数+解\*/

else if (menu == 2 || menu == 3) {

cout << "第" << setw(4) << \_\_stepnumber << " 步(" << setw(2) << number\_of\_plates << "#: "

<< source\_tower << "-->" << destination\_tower << ")";

/\*若菜单项为2，直接在此换行\*/

if (menu == 2)

cout << endl;

}

}

void hanoi\_act\_array(int number\_of\_plates, char source\_tower, char destination\_tower, const int menu)

{

/\*菜单项3、4或8：改变数组值+横式打印内部数组\*/

if (menu == 3 || menu == 4 || menu == 8) {

array\_move\_plates(source\_tower, destination\_tower);

array\_print\_tower(number\_of\_plates, source\_tower, destination\_tower, menu);

}

/\*菜单项4或8：改变数值位置\*/

if (menu == 4 || menu == 8)

array\_show\_move(source\_tower, destination\_tower, menu);

/\*菜单项8：将这一步数组值复位，以便下一步图形化输出能使用移动前的内部数组值\*/

if (menu == 8)

array\_move\_plates(destination\_tower, source\_tower);

}

void array\_tower\_initialize(const int number\_of\_plates, const char source\_tower)

{

int i, j;

/\*给所有元素赋初值\*/

for (i = 0; i < 3; i++) {

for (j = 0; j < PLATE\_NUM\_LIMIT; j++)

TOWER[i][j] = 0;

}

/\*根据起始柱的代号，给起始柱的对应元素赋上1-n的值，代表初始时的各个盘子\*/

for (i = 0; i < number\_of\_plates; i++)

TOWER[source\_tower - 'A'][i] = number\_of\_plates - i;

/\*起始柱栈顶指针的初始化\*/

for (i = 0; i < 3; i++)

MARK[i] = 0;

MARK[source\_tower - 'A'] = number\_of\_plates;

}

void array\_print\_base(const int number\_of\_plates, const char source\_tower, const int menu)

{

int i, BASE\_Y; /\*定义所需变量，BASE\_Y是坐标参数\*/

if (menu == 4)

BASE\_Y = PLATE\_NUM\_LIMIT + 2;

else

BASE\_Y = CONSOLE\_BASE\_Y + PLATE\_NUM\_LIMIT + 2;

/\*循环输出代表三个柱子的字母\*/

for (i = 0; i < 3; i++) {

cct\_gotoxy(ARRAY\_BASE\_X + (ARRAY\_DISTANCE + 1) \* i, BASE\_Y + 1);

cout << char('A' + i);

}

/\*循环输出一排等号，使输出结果更具形象化\*/

cct\_gotoxy(ARRAY\_BASE\_X - 2, BASE\_Y);

for (i = 0; i < 6 + 2 \* ARRAY\_DISTANCE + 1; i++)

cout << "=";

/\*在指定的位置，形象化输出每根柱子上的情况，光标移动前后都需移动到指定位置\*/

cct\_gotoxy((ARRAY\_DISTANCE + 1) \* (source\_tower - 'A') + ARRAY\_BASE\_X - 1, BASE\_Y - 1);

for (i = 0; i < PLATE\_NUM\_LIMIT; i++) {

if (TOWER[source\_tower - 'A'][i])

cout << setw(2) << TOWER[source\_tower - 'A'][i];

cct\_gotoxy((ARRAY\_DISTANCE + 1) \* (source\_tower - 'A') + ARRAY\_BASE\_X - 1, BASE\_Y - i - 2);

}

cct\_gotoxy((ARRAY\_DISTANCE + 1) \* (source\_tower - 'A') + ARRAY\_BASE\_X + 1, BASE\_Y - number\_of\_plates);

}

void array\_show\_move(const char source\_tower, const char destination\_tower, const int menu)

{

int BASE\_Y; /\*定义所需变量，BASE\_Y是坐标参数\*/

/\*按菜单函数返回值决定参数值\*/

if (menu == 4)

BASE\_Y = PLATE\_NUM\_LIMIT + 2;

else

BASE\_Y = CONSOLE\_BASE\_Y + PLATE\_NUM\_LIMIT + 2;

/\*覆盖原有内容\*/

cct\_gotoxy((ARRAY\_DISTANCE + 1) \* (source\_tower - 'A') + ARRAY\_BASE\_X + 1, BASE\_Y - 1 - MARK[source\_tower - 'A']);

cout << "\010\010\040\040";

/\*输出现有内容\*/

cct\_gotoxy((ARRAY\_DISTANCE + 1) \* (destination\_tower - 'A') + ARRAY\_BASE\_X - 1, BASE\_Y - MARK[destination\_tower - 'A']);

cout << setw(2) << TOWER[destination\_tower - 'A'][MARK[destination\_tower - 'A'] - 1];

}

void array\_print\_tower(const int number\_of\_plates, const char source\_tower, const char destination\_tower, const int menu)

{

int i, j, y = 0;

if (menu == 4)

y = PLATE\_NUM\_LIMIT + 7; /\*菜单项4的位置\*/

else if (menu == 8 || menu == 9)

y = CONSOLE\_BASE\_Y + PLATE\_NUM\_LIMIT + 7; /\*菜单项8、菜单项9的位置\*/

if (menu != 3) { /\*菜单项3不需要擦除再输出\*/

cct\_gotoxy(SCREEN\_WIDTH - 1, y);

/\*消除原有内容\*/

for (i = 0; i < SCREEN\_WIDTH; i++)

cout << '\010';

for (i = 0; i < SCREEN\_WIDTH; i++)

cout << '\040';

/\*根据步数决定输出提示内容\*/

cct\_gotoxy(0, y);

if (\_\_stepnumber) /\*已经开始移动，步数不为零，输出移动情况\*/

cout << "第" << setw(4) << \_\_stepnumber << "步(" << number\_of\_plates << "#: " << source\_tower << "-->" << destination\_tower << ") ";

else /\*还未移动，步数为零，输出初始情况\*/

cout << "初始: ";

}

/\*显示内部数组值\*/

for (i = 0; i < 3; i++) {

cout << ' ' << char('A' + i) << ":";

for (j = 0; j < PLATE\_NUM\_LIMIT; j++) {

if (TOWER[i][j] != 0)

cout << setw(2) << TOWER[i][j];

else

cout << " ";

}

}

cout << endl;

}

void array\_move\_plates(const char source\_tower, const char destination\_tower)

{

TOWER[destination\_tower - 'A'][MARK[destination\_tower - 'A']++] = TOWER[source\_tower - 'A'][--MARK[source\_tower - 'A']];

TOWER[source\_tower - 'A'][MARK[source\_tower - 'A']] = 0;

}

void console\_print\_base()

{

const int bg\_color = COLOR\_HYELLOW; //背景为亮黄色

const int fg\_color = COLOR\_HBLUE; //前景为亮蓝色

int i, j; /\*计数变量\*/

/\*光标位置初始化\*/

cct\_gotoxy(CONSOLE\_BASE\_X, CONSOLE\_BASE\_Y);

/\*输出底盘\*/

for (i = 0; i < 3; i++) {

Sleep(50);

cct\_showch(CONSOLE\_BASE\_X + i \* (TOWER\_WIDTH + TOWER\_DISTANCE), CONSOLE\_BASE\_Y, ' ', bg\_color, fg\_color, TOWER\_WIDTH);

}

/\*输出柱子\*/

for (i = 0; i < TOWER\_HEIGHT; i++) {

for (j = 0; j < 3; j++) {

Sleep(50);

cct\_showch(CONSOLE\_BASE\_X + (TOWER\_WIDTH - 1) / 2 + j \* (TOWER\_WIDTH + TOWER\_DISTANCE), CONSOLE\_BASE\_Y - i - 1, ' ', bg\_color, fg\_color, 1);

}

}

/\*输出完成后的颜色重设\*/

cct\_setcolor();

}

void console\_print\_plates(const int number\_of\_plates, const char source\_tower)

{

const int mid = CONSOLE\_BASE\_X + (TOWER\_WIDTH - 1) / 2 + (TOWER\_WIDTH + TOWER\_DISTANCE) \* (source\_tower - 'A'); /\*每根柱子中间的位置\*/

int X, Y; /\*横坐标、纵坐标\*/

int bg\_color, fg\_color; /\*背景色、字体色\*/

int LENGTH; /\*盘子长度\*/

int i = 0; /\*计数变量\*/

/\*输出盘子\*/

for (i = 0; i < number\_of\_plates; i++) {

/\*根据参数，确定各个变量的值\*/

X = mid - number\_of\_plates + i;

Y = CONSOLE\_BASE\_Y - i - 1;

bg\_color = number\_of\_plates - i;

fg\_color = 7;

LENGTH = 2 \* number\_of\_plates + 1 - 2 \* i;

/\*延时、输出\*/

Sleep(30);

cct\_showch(X, Y, ' ', bg\_color, fg\_color, LENGTH);

}

/\*输出完成后的颜色重设\*/

cct\_setcolor();

}

void console\_show\_move(const int number\_of\_plates, const char source\_tower, const char destination\_tower, const int menu)

{

const int bg\_color = number\_of\_plates; /\*背景色（根据盘子编号来取，保证移动过程中盘子不变色）\*/

const int fg\_color = COLOR\_WHITE; /\*字体色\*/

const int LENGTH = 2 \* number\_of\_plates + 1; /\*盘子长度\*/

/\*起始柱中间位置\*/

const int src\_mid = CONSOLE\_BASE\_X + (TOWER\_WIDTH - 1) / 2 + (TOWER\_WIDTH + TOWER\_DISTANCE) \* (source\_tower - 'A');

/\*目标柱中间位置\*/

const int dst\_mid = CONSOLE\_BASE\_X + (TOWER\_WIDTH - 1) / 2 + (TOWER\_WIDTH + TOWER\_DISTANCE) \* (destination\_tower - 'A');

const int START\_X = src\_mid - TOWER[source\_tower - 'A'][MARK[source\_tower - 'A'] - 1]; /\*起始横坐标\*/

const int START\_Y = CONSOLE\_BASE\_Y - MARK[source\_tower - 'A']; /\*起始纵坐标\*/

const int END\_X = dst\_mid - number\_of\_plates; /\*最终横坐标\*/

const int END\_Y = CONSOLE\_BASE\_Y - MARK[destination\_tower - 'A'] - 1; /\*最终纵坐标\*/

int time = 0; /\*延时参数\*/

/\*取延时参数，若为菜单项7、8中的单步演示，延时100；若为菜单项8中的连续演示，延时delay\_time；若为菜单项9，延时1，造成瞬移效果\*/

if (menu == 7 || (menu == 8 && \_\_delay\_time == 0))

time = 100;

else if (menu == 8 && \_\_delay\_time != 0)

time = \_\_delay\_time;

else if (menu == 9)

time = 1;

/\*向上移动\*/

console\_show\_move\_up(START\_X, START\_Y, bg\_color, fg\_color, src\_mid, LENGTH, time);

/\*左右移动\*/

if (destination\_tower < source\_tower)

console\_show\_move\_left(START\_X, END\_X, bg\_color, fg\_color, LENGTH, time);

else

console\_show\_move\_right(START\_X, END\_X, bg\_color, fg\_color, LENGTH, time);

/\*向下移动\*/

console\_show\_move\_down(END\_X, END\_Y, bg\_color, fg\_color, dst\_mid, LENGTH, time);

/\*移动完成后的颜色重设\*/

cct\_setcolor();

}

void console\_show\_move\_up(int START\_X, int START\_Y, int bg\_color, int fg\_color, int src\_mid, int LENGTH, int time)

{

int Y; /\*显示动画中变动的当前纵坐标\*/

for (Y = START\_Y; Y >= 1; Y--) {

cct\_showch(START\_X, Y, ' ', bg\_color, fg\_color, LENGTH);

/\*若连续演示速度不为5，则启动延时\*/

if (time != 1)

Sleep(time);

/\*到达最高点之前，用标准色擦除原有内容，造成动画效果\*/

if (Y > 1)

cct\_showch(START\_X, Y, ' ', COLOR\_BLACK, COLOR\_WHITE, LENGTH);

/\*若未离开柱子，则需在柱子中间位置重新显示柱子，否则会被上一句擦除掉\*/

if (Y > TOWER\_TOP)

cct\_showch(src\_mid, Y, ' ', COLOR\_HYELLOW, COLOR\_WHITE, 1);

}

}

void console\_show\_move\_down(int END\_X, int END\_Y, int bg\_color, int fg\_color, int dst\_mid, int LENGTH, int time)

{

int Y; /\*显示动画中变动的当前纵坐标\*/

for (Y = 1; Y <= END\_Y; Y++) {

cct\_showch(END\_X, Y, ' ', bg\_color, fg\_color, LENGTH);

/\*若连续演示速度不为5，则启动延时\*/

if (time != 1)

Sleep(time);

/\*到达最低点之前，用标准色擦除原有内容，造成动画效果\*/

if (Y < END\_Y)

cct\_showch(END\_X, Y, ' ', COLOR\_BLACK, COLOR\_WHITE, LENGTH);

/\*若进入柱子，则需在柱子中间位置重新显示柱子，否则会被上一句擦除掉\*/

if (Y > TOWER\_TOP && Y < END\_Y)

cct\_showch(dst\_mid, Y, ' ', COLOR\_HYELLOW, COLOR\_WHITE, 1);

}

}

void console\_show\_move\_left(int START\_X, int END\_X, int bg\_color, int fg\_color, int LENGTH, int time)

{

int X; /\*显示动画中变动的当前横坐标\*/

for (X = START\_X; X >= END\_X; X--) {

cct\_showch(X, 1, ' ', bg\_color, fg\_color, LENGTH);

/\*若连续演示速度不为5，则启动延时\*/

if (time != 1)

Sleep(time);

/\*到达最左侧之前，用标准色擦除原有内容，造成动画效果\*/

if (X > END\_X)

cct\_showch(X, 1, ' ', COLOR\_BLACK, COLOR\_WHITE, LENGTH);

}

}

void console\_show\_move\_right(int START\_X, int END\_X, int bg\_color, int fg\_color, int LENGTH, int time)

{

int X; /\*显示动画中变动的当前横坐标\*/

for (X = START\_X; X <= END\_X; X++) {

cct\_showch(X, 1, ' ', bg\_color, fg\_color, LENGTH);

/\*若连续演示速度不为5，则启动延时\*/

if (time != 1)

Sleep(time);

/\*到达最右侧之前，用标准色擦除原有内容，造成动画效果\*/

if (X < END\_X)

cct\_showch(X, 1, ' ', COLOR\_BLACK, COLOR\_WHITE, LENGTH);

}

}

void game\_command\_center(const int total\_number\_of\_plates, const char destination\_tower, const int menu)

{

cct\_setcursor(CURSOR\_VISIBLE\_NORMAL); /\*显示光标\*/

cct\_gotoxy(0, GAME\_TIP\_Y); /\*移动光标至指定位置\*/

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) ："; /\*输出提示语。改变提示语长度时，请同步改变#define TIP\_LENGTH\*/

while (1) {

int i = 0; /\*计数变量，每一次循环结束归零\*/

char src = 0, dst = 0, c; /\*起始柱编号（每一次循环结束归零）、目标柱编号（每一次循环结束归零）、提取字符\*/

/\*1、输入指令\*/

while (1) {

c = getchar(); /\*逐个提取字符\*/

if (c == '\n' || c == 'q' || c == 'Q') /\*若有换行符或退出指令，结束输入\*/

break;

if (i == 0) /\*若输入未结束，第一个字符赋给src\*/

src = c;

if (i == 1) /\*若输入未结束，第二个字符赋给dst\*/

dst = c;

i++; /\*每输入一个字符，计数变量自增，统计输入的字符数\*/

}

/\*2、若c为退出指令，输出提示语并结束游戏\*/

if (c == 'q' || c == 'Q') {

cout << "游戏中止!!!!!";

break;

}

/\*3、若刚好输入两个不等的字符，调整大小写，判断是否可以移动并移动盘子\*/

if (i == 2 && src != dst) {

if (src >= 'a' && src <= 'c')

src = src - 'a' + 'A';

if (dst >= 'a' && dst <= 'c')

dst = dst - 'a' + 'A';

game\_get\_info\_and\_move(src, dst, menu);

}

/\*4、若1号盘已经到达最终位置，则判断游戏结束\*/

if (TOWER[destination\_tower - 'A'][total\_number\_of\_plates - 1] == 1) {

cct\_gotoxy(0, GAME\_TIP\_Y + 1);

cout << "游戏结束!!!!!";

break;

}

/\*5、输入指令错误，或者是每完成一步移动后，清除原有指令（位置靠后，不会在游戏结束或错误提示时先清空原有指令）\*/

else

game\_clear\_input();

}

}

void game\_clear\_input()

{

int j;

cct\_gotoxy(TIP\_LENGTH, GAME\_TIP\_Y);

for (j = 0; j < 60; j++)

cout << '\040';

cct\_gotoxy(TIP\_LENGTH, GAME\_TIP\_Y);

}

void game\_get\_info\_and\_move(const char source\_tower, const char destination\_tower, const int menu)

{

/\*根据柱子编号，取栈顶元素的值（空为零，错误为负）\*/

int number\_of\_src = game\_get\_plate\_num(source\_tower);

int number\_of\_dst = game\_get\_plate\_num(destination\_tower);

/\*只有当两个栈顶元素值都不为负（都正确）的时候，才进行下一步判断\*/

if (number\_of\_src >= 0 && number\_of\_dst >= 0) {

/\*起始柱为空的情况\*/

if (TOWER[source\_tower - 'A'][0] == 0)

game\_err(EmptySourceTower);

/\*大盘压小盘的情况\*/

else if (number\_of\_src > number\_of\_dst && number\_of\_dst != 0)

game\_err(BigOnSmall);

/\*准备就绪，可以移动的情况\*/

else

game\_ok\_move(number\_of\_src, source\_tower, destination\_tower, menu);

}

}

int game\_get\_plate\_num(const char tower)

{

/\*柱子编号范围正确\*/

if (tower >= 'A' && tower <= 'C') {

/\*使用只读字符变量接收参数tower，消除智能提示\*/

const char Tower = tower;

/\*若栈底元素不为零，说明该柱不为空，返回栈顶元素的值\*/

if (TOWER[Tower - 'A'][0])

return TOWER[Tower - 'A'][MARK[Tower - 'A'] - 1];

/\*若栈底元素为零，说明该柱为空，返回零\*/

else

return 0;

}

/\*若柱子编号不正确，返回负数\*/

else

return -1;

}

void game\_err(const int err\_info)

{

unsigned int i; /\*计数变量\*/

char c[128]; /\*接收提示语的字符数组\*/

/\*根据错误信息制定提示语\*/

if (err\_info == 1)

strcpy(c, "源柱为空!");

else if (err\_info == 2)

strcpy(c, "大盘压小盘，非法移动!");

/\*输出提示语并暂停\*/

cout << c;

Sleep(1000);

/\*消除提示语并使光标复位\*/

cct\_gotoxy(0, GAME\_TIP\_Y + 1);

for (i = 0; i < strlen(c); i++)

cout << '\040';

cct\_gotoxy(TIP\_LENGTH, GAME\_TIP\_Y);

}

void game\_ok\_move(const int number\_of\_plates, const char source\_tower, const char destination\_tower, const int menu)

{

\_\_stepnumber++; /\*步数自增\*/

array\_move\_plates(source\_tower, destination\_tower); /\*改变数组值，内部数组达到移动盘子的效果\*/

array\_print\_tower(number\_of\_plates, source\_tower, destination\_tower, menu); /\*横式打印内部数组\*/

array\_show\_move(source\_tower, destination\_tower, menu); /\*改变数值位置，纵向输出达到移动盘子的效果\*/

array\_move\_plates(destination\_tower, source\_tower); /\*将这一步数组值复位，以便下一步图形化输出能使用移动前的内部数组值\*/

console\_show\_move(number\_of\_plates, source\_tower, destination\_tower, menu); /\*改变盘子位置，图形输出达到移动盘子的效果\*/

array\_move\_plates(source\_tower, destination\_tower); /\*改变数组值，内部数组达到移动盘子的效果\*/

}

void input\_info(int\* number\_of\_plates, char\* source\_tower, char\* destination\_tower, int\* delay\_time\_set, const int menu)

{

if (menu != 5) { /\*菜单项5不需要输入信息\*/

input\_number\_of\_plates(number\_of\_plates);

input\_source\_tower(source\_tower);

input\_destination\_tower(destination\_tower, source\_tower);

}

if (menu == 4 || menu == 8) /\*菜单项4和菜单项8需要输入延时参数\*/

input\_delay(delay\_time\_set);

}

void input\_number\_of\_plates(int\* number\_of\_plates)

{

int n;

while (1) {

cout << "请输入汉诺塔的层数(1-" << PLATE\_NUM\_LIMIT << ")：" << endl;

cin >> n;

if (n >= 1 && n <= PLATE\_NUM\_LIMIT && cin.good() == 1) {

cin.clear();

cin.ignore(INT\_MAX, '\n');

break;

}

/\*只剩非法输入，或数值超类型范围的处理\*/

else if ((n < 1 || n > PLATE\_NUM\_LIMIT) && cin.good() == 0) {

cin.clear();

cin.ignore(INT\_MAX, '\n');

}

/\*读取状态正常，超过用户范围时的处理\*/

else if ((n < 1 || n > PLATE\_NUM\_LIMIT) && cin.good() == 1) {

cin.clear();

cin.ignore(INT\_MAX, '\n');

}

}

\*number\_of\_plates = n;

}

void input\_source\_tower(char\* source\_tower)

{

char src;

while (1) {

cout << "请输入起始柱(A-C)：" << endl;

cin >> src;

if (((src >= 'a' && src <= 'c') || (src >= 'A' && src <= 'C')) && cin.good() == 1) {

cin.clear();

cin.ignore(INT\_MAX, '\n');

break;

}

/\*只剩非法输入，或数值超类型范围的处理\*/

else if (((src < 'a' || src > 'c') || (src < 'A' || src > 'C')) && cin.good() == 0) {

cin.clear();

cin.ignore(INT\_MAX, '\n');

}

/\*读取状态正常，超过用户范围时的处理\*/

else if (((src < 'a' || src > 'c') || (src < 'A' || src > 'C')) && cin.good() == 1) {

cin.clear();

cin.ignore(INT\_MAX, '\n');

}

}

if (src >= 'a' && src <= 'c') {

\*source\_tower = src - 32;

}

else {

\*source\_tower = src;

}

}

void input\_destination\_tower(char\* destination\_tower, const char\* source\_tower)

{

char dst;

while (1) {

cout << "请输入目标柱(A-C)：" << endl;

cin >> dst;

if (((dst >= 'a' && dst <= 'c') || (dst >= 'A' && dst <= 'C')) && cin.good() == 1) {

if (dst != \*source\_tower && dst != \*source\_tower + 32 && dst != \*source\_tower - 32) {

cin.clear();

cin.ignore(INT\_MAX, '\n');

break;

}

/\*dst与src相等时的处理\*/

else {

/\*为了使提示语中的dst输出为大写，对dst的值进行处理\*/

if (dst >= 'a' && dst <= 'c') {

dst = dst - 32;

}

cout << "目标柱(" << dst << ")不能与起始柱(" << \*source\_tower << ")相同" << endl;

cin.clear();

cin.ignore(INT\_MAX, '\n');

}

}

/\*只剩非法输入，或数值超类型范围的处理\*/

else if (((dst < 'a' || dst > 'c') || (dst < 'A' || dst > 'C')) && cin.good() == 0) {

cin.clear();

cin.ignore(INT\_MAX, '\n');

}

/\*读取状态正常，超过用户范围时的处理\*/

else if (((dst < 'a' || dst > 'c') || (dst < 'A' || dst > 'C')) && cin.good() == 1) {

cin.clear();

cin.ignore(INT\_MAX, '\n');

}

}

if (dst >= 'a' && dst <= 'c') {

\*destination\_tower = dst - 32;

}

else {

\*destination\_tower = dst;

}

}

void input\_delay(int\* delay\_time\_set)

{

int i;

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 >> i;

if (i >= 0 && i <= 5 && cin.good() == 1) {

cin.clear();

cin.ignore(INT\_MAX, '\n');

break;

}

/\*只剩非法输入，或数值超类型范围的处理\*/

else if (cin.good() == 0) {

cin.clear();

cin.ignore(INT\_MAX, '\n');

}

/\*读取状态正常，超过用户范围时的处理\*/

else if ((i < 0 || i > 5) && cin.good() == 1) {

cin.clear();

cin.ignore(INT\_MAX, '\n');

}

}

/\*根据i的输入值，给全局变量delay\_time赋值\*/

if (i == 0)

\_\_delay\_time = 0;

else if (i >= 1 && i <= 3)

\_\_delay\_time = 130 - 30 \* i;

else if (i == 4)

\_\_delay\_time = 15;

else if (i == 5)

\_\_delay\_time = 1;

\*delay\_time\_set = i;

}

void pause(const int delay\_time\_set)

{

int c;

if (delay\_time\_set > 0 && delay\_time\_set != 5)

Sleep(\_\_delay\_time);

else {

while (1) {

c = \_getch();

if (c == '\r')

break;

}

}

}

void reset(const int menu)

{

/\*菜单项1-3输出完成后不需要移动光标，输出换行符即可\*/

if (menu >= 1 && menu <= 3)

cout << endl;

/\*重置颜色\*/

cct\_setcolor();

/\*菜单项4-9输出完成后需要移动光标，根据菜单函数返回值调整参数\*/

if (menu >= 4 && menu <= 8)

cct\_gotoxy(0, CONSOLE\_BASE\_Y + PLATE\_NUM\_LIMIT + 9);

else if (menu == 9)

cct\_gotoxy(0, CONSOLE\_BASE\_Y + PLATE\_NUM\_LIMIT + 13);

cout << "按回车键继续"; /\*输出提示语\*/

\_\_stepnumber = 0; /\*重置记录步数的静态全局变量\*/

\_\_delay\_time = 0; /\*重置标志延时时长的静态全局变量\*/

pause(0); /\*所有重置项完成后的中断\*/

cct\_cls(); /\*清理屏幕\*/

}

void command\_center(int number\_of\_plates, char source\_tower, char temporary\_tower, char destination\_tower, int delay\_time\_set, int menu)

{

/\*清空屏幕\*/

if (menu >= 4 && menu <= 9)

cct\_cls();

/\*内部数组初始化\*/

if (menu == 3 || menu == 4 || menu == 7 || menu == 8 || menu == 9)

array\_tower\_initialize(number\_of\_plates, source\_tower);

command\_info(number\_of\_plates, source\_tower, destination\_tower, delay\_time\_set, menu); /\*输出提示语\*/

command\_array\_base(number\_of\_plates, source\_tower, destination\_tower, menu); /\*输出纵向数组\*/

command\_console\_base(number\_of\_plates, source\_tower, menu); /\*输出柱子和盘子\*/

command\_delay(menu); /\*控制中断或延时\*/

/\*调用递归函数\*/

if (menu != 5 && menu != 6 && menu != 9)

hanoi(number\_of\_plates, source\_tower, temporary\_tower, destination\_tower, menu);

/\*运行游戏\*/

if (menu == 9)

game\_command\_center(number\_of\_plates, destination\_tower, menu);

}

void command\_info(const int number\_of\_plates, const char source\_tower, const char destination\_tower, const int delay\_time\_set, const int menu)

{

if (menu == 4 || menu == 6 || menu == 7 || menu == 8 || menu == 9)

cout << "从 " << source\_tower << " 移动到 " << destination\_tower << "，共 " << number\_of\_plates << " 层";

if (menu == 4 || menu == 8)

cout << " ，延时设置为 " << delay\_time\_set;

}

void command\_array\_base(const int number\_of\_plates, const char source\_tower, const char destination\_tower, const int menu)

{

if (menu == 4 || menu == 8 || menu == 9) {

array\_print\_tower(number\_of\_plates, source\_tower, destination\_tower, menu);

array\_print\_base(number\_of\_plates, source\_tower, menu);

}

}

void command\_console\_base(const int number\_of\_plates, const char source\_tower, const int menu)

{

if (menu >= 5 && menu <= 9)

console\_print\_base();

if (menu >= 6 && menu <= 9)

console\_print\_plates(number\_of\_plates, source\_tower);

}

void command\_delay(const int menu)

{

if (menu == 4 || menu == 8)

pause(\_\_delay\_time);

else if (menu == 7)

Sleep(1000);

else if (menu == 9)

Sleep(100);

}

**2.hanoi\_main.cpp**

/\*2151294 信11 马威\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <Windows.h>

#include "cmd\_console\_tools.h"

#include "hanoi.h"

using namespace std;

int main()

{

cct\_setconsoleborder(120, 40, 120, 9000);

while (1) {

int MENU = menu();

cout << MENU << "\n\n\n";

Sleep(200);

if (MENU == 0)

break;

int number\_of\_plates, \* nop = &number\_of\_plates;

int delay\_time\_set = 0, \* dts = &delay\_time\_set;

char src\_tower, \* st = &src\_tower;

char dst\_tower, \* dt = &dst\_tower;

input\_info(nop, st, dt, dts, MENU);

char tmp\_tower = 'A' + 'B' + 'C' - src\_tower - dst\_tower;

cct\_setcursor(CURSOR\_INVISIBLE);

command\_center(number\_of\_plates, src\_tower, tmp\_tower, dst\_tower, delay\_time\_set, MENU);

cct\_setcursor(CURSOR\_VISIBLE\_NORMAL);

reset(MENU);

}

return 0;

}

**3.hanoi\_menu.cpp**

/\*2151294 信11 马威\*/

#include<iostream>

#include<conio.h>

using namespace std;

int menu()

{

int ret;

cout << "---------------------------------\n"

<< "1.基本解\n"

<< "2.基本解(步数记录)\n"

<< "3.内部数组显示(横向)\n"

<< "4.内部数组显示(纵向+横向)\n"

<< "5.图形解-预备-画三个圆柱\n"

<< "6.图形解-预备-在起始柱上画n个盘子\n"

<< "7.图形解-预备-第一次移动\n"

<< "8.图形解-自动移动版本\n"

<< "9.图形解-游戏版\n"

<< "0.退出\n"

<< "---------------------------------\n"

<< "[请选择:] ";

while (1) {

ret = \_getch();

if (ret >= '0' && ret <= '9') {

break;

}

}

return ret - '0';

}