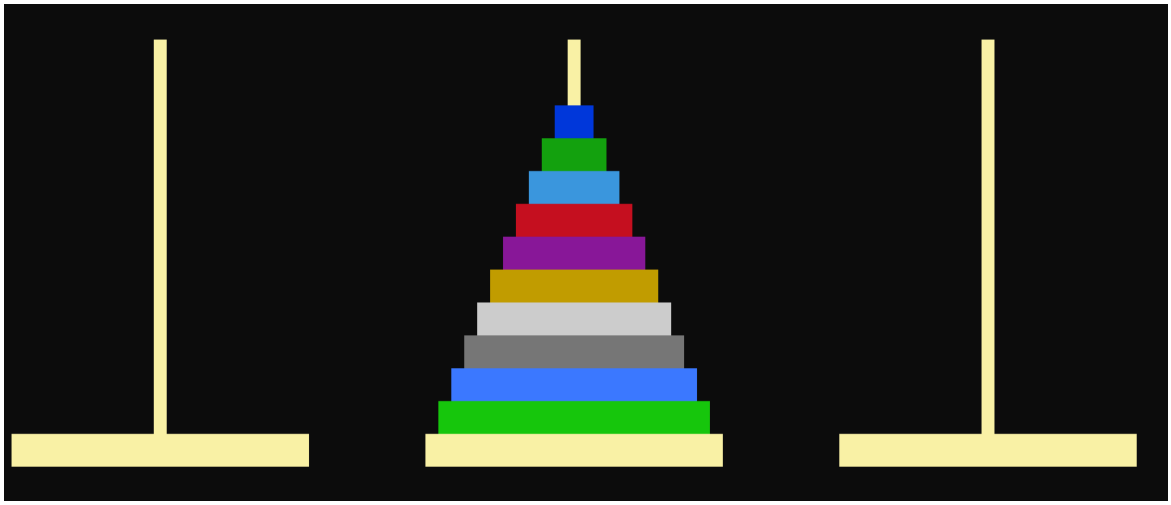
同济大学 实验报告

报告名称：汉诺塔综合报告

装 订 线

高程1班 **信09**

1953729 **吴浩泽**



完成日期：2020年12月1日

1. 题目

将之前做的所有汉诺塔的各小题集成在一个程序中，用菜单方式进行选择，并加入图形化演示的要求

菜单一

求汉诺塔基本解

菜单二

求汉诺塔基本解并给出步数

菜单三

横向输出，显示内部数组

菜单四

横向和竖向输出，显示内部数组

菜单五

在屏幕上画出三根圆柱

菜单六

在屏幕上画出三根圆柱，并画出n个圆盘

菜单七

在六基础上完成一次移动

菜单八

整个汉诺塔横向竖向输出，显示内部数组，并显示对应图形变化，盘子的移动

菜单九

汉诺塔游戏，人工移动盘子，不再一次性移动整个过程

2. 整体设计思路

从头开始想这道题，肯定是有难度的，那么可以一步一步去分析。整道题是积少成多，不是一次就能都想出来的，所以将其分块才是最正确的处理方式。

main函数里是比较简单的，过程清晰明了。

首先先确定cmd窗口大小。

之后调用了menu菜单函数，每次选项结束后都会重新清屏并重新罗列菜单。

接下来就是具体的格式，控制了输入，而因为发现菜单12346789前面都是同样的类型，所以考虑功能一样的放在一个函数调用，而具体的差异就会在里面细分。

而输入完成后接下来可以具体分步操作，首先是递归，12348项应用到递归，所以同样是考虑到过程类似，整合成一个递归函数，然后差异通过参数的不同调用不同的部分。

接下来再看56789都用到了图形化，也就是所谓的盘子，所以可以归结为一个函数，我把它命名为draw函数，具体差异仍然通过参数的传递而选择不同的方式。

56仅仅是画图，没有移动的过程，也没有递归，那么在主函数里把他们两个拿出来。

789都有盘子移动的过程，那么再创建个函数是负责移动盘子，我将其命名为move函数

具体再分的时候因为9这一项包含的比较复杂，所以我可以单独建一个函数，我命名为ninth函数，这个函数就是为了完成9这一项的功能的。

整个过程是在不断循环的，所以我把整个main函数用一个while(1)包括起来，知道输入时输入了0，菜单0项才退出循环，整个程序结束。

接下来大致介绍各个函数的设计思路

menu函数顾名思义，就是菜单函数，是为了完成菜单选项，并且给出返回值，再在main函数里根据菜单函数的返回值确定以后的每一步应该调用什么函数。

hanoi函数是整个项目里唯一的递归函数，它的作用就是递归来正确完成程序的执行，而n等于1和n-1的情况下，两部分是一样的，那么可以把这两部分再抽出一个函数，来选择每一步该用什么函数，就有了choose函数。

choose函数是选择递归函数里应该用哪个函数，并且因为hanoi函数的行数限制，所以把核心的递归代码放到了choose中，也就是下面这一部分。

```
if (src == 'A' && dst == 'B')
    b[top2++] = a[--top1];
else if (src == 'A' && dst == 'C')
    c[top3++] = a[--top1];
else if (src == 'B' && dst == 'C')
    c[top3++] = b[--top2];
else if (src == 'B' && dst == 'A')
    a[top1++] = b[--top2];
else if (src == 'C' && dst == 'A')
    a[top1++] = c[--top3];
else
    b[top2++] = c[--top3];
```

其中a[],b[],c[],对应的top1,top2,top3,是静态全局变量，用来具体实现递归过程的。

接下来choose函数要分步选择该调用哪个函数，那么就会分为第一个函数，第二个函数，横向函数，纵向函数，第四个函数，第八个函数。

first函数比较简单，第一个函数只负责在每个小过程上输出一句话，所以整个函数只有一

句话就可以了。

同样的第二个函数second函数也是一样，需要记录步数，定义一个静态全局变量，来记录步数，输出也是只需要输出一句话就可以了。

接下来是第三个函数，而第三个函数就是横向输出函数，为了后面函数也用到，这里就将其写成heng函数，横向输出的意思。

第四个函数，是横向和竖向的结合，那么就要包括第三个函数，也就是包括heng函数，而自己独立又要需要竖向输出函数，那么很自然联想到再建一个shu函数，然后这个第四个函数需要有延时操作，那么也是用一个静态全局变量来控制延时问题。

这里说一下shu函数，就是为了竖向输出，以后也会用到这个，所以就将其单独拿出来作为一个函数。shu函数因为位置输出会有差异，所以会比heng函数多出一个参数，来专门控制输出位置，分类时，4是一个单独的，8和9的位置是一样的。

接下来就是第五个画柱子的函数，那么可以单独定义一个draw函数，虽然画柱子是一样的，但是为了方便，我在draw函数里也通过参数选择进行了其他操作，当然了都是些细小差异，不值一提。

通过画柱子会发现，还有画盘子，而画盘子就可以被画柱子调用，这样参数选择就用上了。

我们再来看输入函数input函数，这个函数比较大，里面放了各种输入错误处理等，并进行了一些元素的赋值。然后通过参数选择，又处理了不同参数对应的细节输入处理，比如延时等的处理。而后，写的是横向输出的初始打印，以及纵向输出的初始打印。

接下来剩的没有解决的只有第九个函数了，而在此之前先说一下其余小的函数

end函数我的目的是输出“按回车键继续”，因为它的456789的位置都是固定的，123的位置是在之后换两行输出，所以可以写出一个end函数控制结束输出。

还有一个是move函数，这也是核心，是移动色块的函数，看似是移动了整个色块，实际上是打印的空格，控制不同参数，达到不同颜色的目的。这个函数同样参数非常多，因为需要分类，一共分了六类，具体靠初始src，目标dst以及移动的过程。而src和dst是分为A B,A C,B A,B C,C A,C B六种情况，而每一种情况的移动都是分为向上，中间移动，和向下移动。这样就可以实现move函数的功能。

接下来就是第九个函数的实现了，第九个函数之所以难，是因为输入的问题，实现上其实没那么费劲，关键是他前面的输入，而一旦输入的符合条件，接下来每一步都进行heng，shu，以及move的函数调用，就可以实现控制一步一动的标准，而终止条件就是所有的色块都移动到目标柱，就可以停止。

所以综上，整个程序设计思路大致明确，每个函数的分工，main函数的调用，都是一步一步思路清晰的，实现起来其实就是细节多，框架有了，实现就不难了。

3. 主要功能的实现

和上一点有些略有重复，在此处就不赘述了

主要讲一下特别关键的函数的几点，简单函数就先不说了。

首先是递归函数的做法是一样的

hanoi的函数过程就是这样

```
void hanoi(int n, char src, char tmp, char dst, int num)                    //    递归函数
{
    if (n == 1)
        choose(n, src, tmp, dst, num);
    else
    {
        hanoi(n - 1, src, dst, tmp, num);
        choose(n, src, tmp, dst, num);
        hanoi(n - 1, tmp, src, dst, num);
    }
}
```

非常核心，全程最关键的递归函数的过程。

对应的choose函数如下：

```
void choose(int n, char src, char tmp, char dst, int num)   //  选择递归函数里应该用哪个函数
{
    if (src == 'A' && dst == 'B')
        b[top2++] = a[--top1];
    else if (src == 'A' && dst == 'C')
        c[top3++] = a[--top1];
    else if (src == 'B' && dst == 'C')
        c[top3++] = b[--top2];
    else if (src == 'B' && dst == 'A')
        a[top1++] = b[--top2];
    else if (src == 'C' && dst == 'A')
        a[top1++] = c[--top3];
    else
        b[top2++] = c[--top3];
    if (num == 1)
        first(n, src, dst);
    else if (num == 2)
        second(n, src, dst);
    else if (num == 3)
        heng(n, src, dst);
    else if (num == 4)
        fourth(n, src, dst, num);
    else if (num == 8)
        eighth(n, src, dst, num);
}
```

每一个条件对应选择哪个函数非常明确。

画柱子的过程如下：

具体是淡黄色的柱子

```
    cct_setcursor(3);
    cct_showch(1, 15, ' ', 14, 0, 23);
    cct_showch(33, 15, ' ', 14, 0, 23);
    cct_showch(65, 15, ' ', 14, 0, 23);
    for (int y = 14; y >= 3; y--)
    {
        Sleep(50);
        cct_showch(12, y, ' ', 14, 0, 1);
        cct_showch(44, y, ' ', 14, 0, 1);
        cct_showch(76, y, ' ', 14, 0, 1);
    }
```

画盘子就不赘述了，也是一个函数，只不过画柱子改成了画盘子。

输入函数和之前的大概思想是一样的，差别就是以后可能会调用，所以要通过形参而改变实参，那么里面都是传递的指针，来改变对应的值。

按回车键结束的函数如下：

```
int end(int num)                                                    //    按回车键继续
{
    x = 1;
    if (num == 4 || num == 5 || num == 6 || num == 7 || num == 8 || num == 9)
        cct_gotoxy(0, 37);
    cout << endl;
    cct_setcursor(2);
    cct_setcolor(0, 7);
    cout << "按回车键继续";
    while (_getch() != '\r')
        ;
    cct_cls();
    num = menu();
    return num;
}
```

注意对应的光标的位置

最重点的还是move函数的移动

重点是分类的思想和移动的过程，是从下到上，横向平移，再从上到下，每一个步骤都需要细致，马虎一点就可能实现有问题。

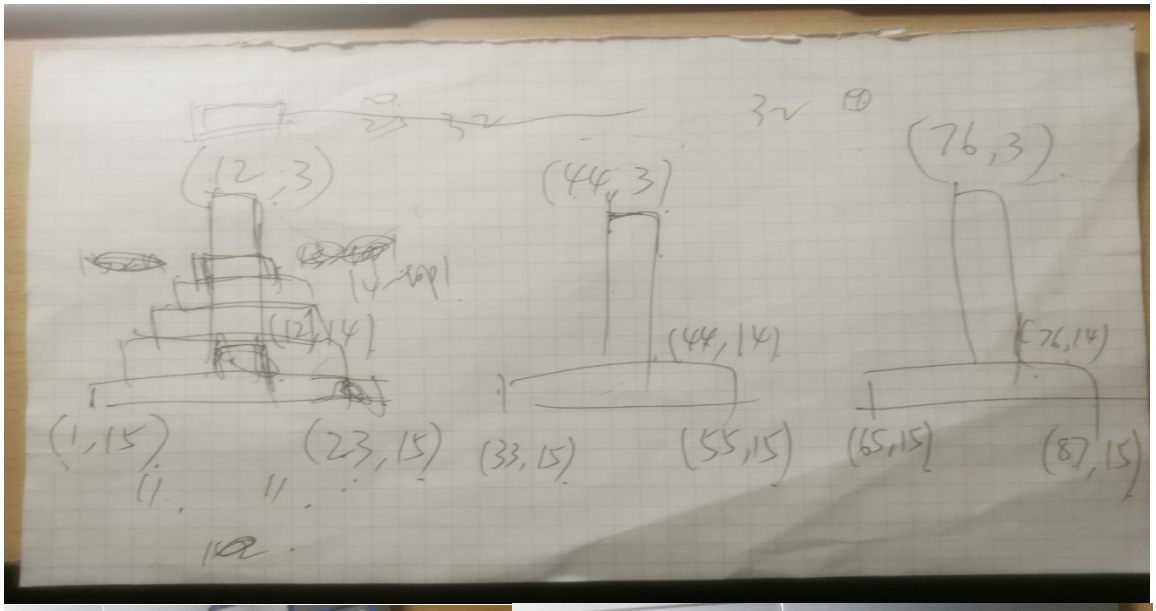
最难的还是第九个函数的实现

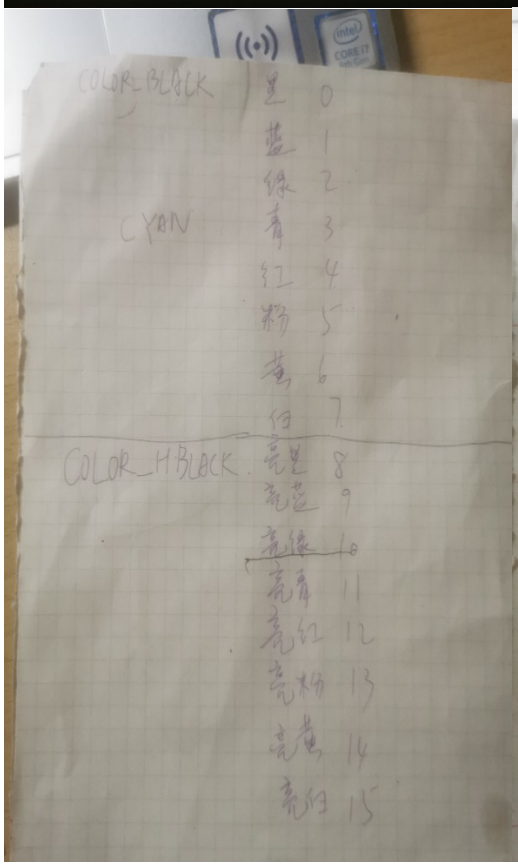
第九个函数刚开始的输入就很难，我也是花了相当长的时间完成了demo中的细节，

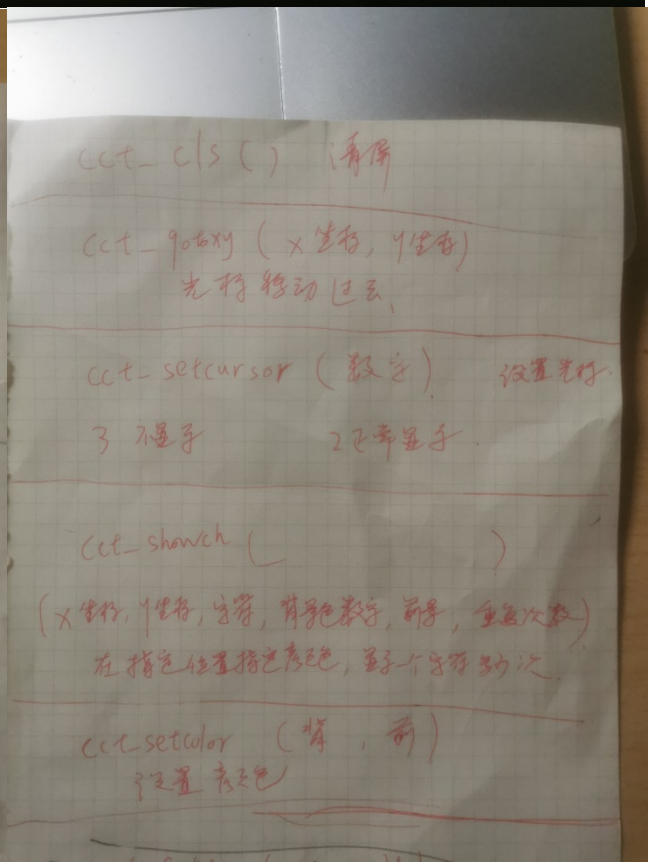
源柱为空的分类，大盘压小盘的分类，以及各种错误输入的问题，控制输入20个自动重新输入等一系列问题都非常细致，不那么容易达到预期标准的。

还有如果每一步输q的话中止以及最后完成标准会结束，提示游戏结束等都是小细节。

下面列举了整个程序用到的函数以及色块颜色，以及各个汉诺塔的坐标，这样一看函数用的东西就很明确了







装 订 线

4. 调试过程碰到的问题

1.第九个菜单输入的问题

这个过程很费劲的，刚开始选择了cin输入，然后清除缓冲区的办法，但是这样的话，输入回车就直接换行到下面去了，不可取，而且也不正确，那么果断考虑getche，因为是有回显的，然后又考虑到赋值问题，直接想到再建一个循环，然后建字符数组，之后循环出来再赋给src和tmp值，后来再判断循环内部的一系列小问题。

```
while (!(i == 3 && whz[2] == '\r' || i == 2 && (whz[0] == 'q' || whz[0] == 'Q') && whz[1] == '\r'))
        {
            if (i == 20)
            {
                cct_showch(60, 34, ' ', 0, 7, 20);
                cct_gotoxy(60, 34);
                i = 0;
            }
            whz[i] = _getche();
            if (whz[i] == ' ')
            {
                cout << '\b';
                i--;
            }
            if (whz[i] == '\b')
            {
                if (i == 0)
                    cout << ' ';
                else
                    cout << whz[i - 1];
                i--;
            }
            i++;
if (!(i == 3 && whz[2] == '\r' || i == 2 && (whz[0] == 'q' || whz[0] == 'Q') && whz[1] == '\r')
&& whz[i - 1] == '\r')
            {
                cct_showch(60, 34, ' ', 0, 7, 20);
                cct_gotoxy(60, 34);
                i = 0;
            }
        }
        src = whz[0];
        tmp = whz[1];
```

2.move函数的处理与top1,top2,top3和全局数组的关系

感觉这个并不太像是一个单纯的逻辑问题，可能需要一定数学发现，我也是找一些规律，观察数字的联系，建立了图形和数组以及栈顶指针的联系，

```
        for (y = 14 - top1; y > 1; y--)
        {
            cct_showch(x - a[top1], y, ' ', 0, 0, a[top1]);
            if (y < 3)
                cct_showch(x, y, ' ', 0, 0, 1);
            else
```

```
            cct_showch(x, y, ' ', 14, 0, 1);
        cct_showch(x + 1, y, ' ', 0, 0, a[top1]);
        cct_showch(x - a[top1], y - 1, ' ', a[top1], 0, 2 * a[top1] + 1);
        if (t != 0)
            Sleep(t);
        else
            Sleep(30);
    }
```

其中的一些运算，都是数字之间的联系。

5. 心得体会

本次作业很大，内容很多，收获也很多，而且非常锻炼逻辑思维，为以后的逻辑编程打下了基础，那么我来具体说说都增长了哪些知识。

在这种较为复杂的程序中，同一个功能一定要分函数来实现，而且函数的细微差别可以靠参数的不同而改变。而且一定要统筹兼顾，做前面考虑后面，要尽量使后面的能用到前面的，在初始写的时候给参数留下空间，以便于后面加进来，改动不会太大，方便后续操作。

本次作业的输入参数调用函数，移动色块函数，递归调用，分情况调用的函数，以及画柱子，画盘子等，都可以分出类别，划分明确。

6. 附件：源程序

```
int main()
{
    cct_setconsoleborder(120, 40, 120, 9000);
/* 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）*/
    int n;
    char src, dst, tmp;
    int* pn = &n;
    char* ps = &src, * pt = &tmp, * pd = &dst;
    int num = menu();
    cout << endl << endl;
    while (1)
    {
        input(pn, ps, pt, pd, num);
        if (num == 0)
        {
            cout << endl << endl;
            break;
        }
        if (num == 1 || num == 2 || num == 3 || num == 4 || num == 8)
            hanoi(n, src, tmp, dst, num);
        else if (num == 5 || num == 6)
        {
            cct_cls();
            draw(n, src, dst, num);
        }
        else if (num == 7)
        {
            cct_cls();
            if (n % 2 != 0)
                draw(n, src, dst, num);
            else
                draw(n, src, tmp, num);
            Sleep(1000);
            if (n % 2 != 0)
                move(n, src, dst, num);
            else
                move(n, src, tmp, num);
        }
        else if (num == 9)
            ninth(n, src, tmp, dst, num);
        num = end(num);
        cout << endl << endl;
    }
    return 0;
}


int menu()
{
    cout <<
"--------------------------------" << endl;
```

```
void shu(int n, char src, char dst, int num)
//    竖向输出
{
    int ch;
    if (num == 4)
    {
        if (src == 'A')
        {
            ch = 10;
            cct_gotoxy(ch, 11 - top1);
            cout << ' ';
            cct_gotoxy(ch + 1, 11 - top1);
            cout << ' ';
        }
        else if (src == 'B')
        {
            ch = 20;
            cct_gotoxy(ch, 11 - top2);
            cout << ' ';
            cct_gotoxy(ch + 1, 11 - top2);
            cout << ' ';
        }
        else
        {
            ch = 30;
            cct_gotoxy(ch, 11 - top3);
            cout << ' ';
            cct_gotoxy(ch + 1, 11 - top3);
            cout << ' ';
        }

        if (dst == 'A')
        {
            ch = 10;
            cct_gotoxy(ch, 12 - top1);
            cout << setw(2) << n;
        }
        else if (dst == 'B')
        {
            ch = 20;
            cct_gotoxy(ch, 12 - top2);
            cout << setw(2) << n;
        }
        else
        {
            ch = 30;
            cct_gotoxy(ch, 12 - top3);
            cout << setw(2) << n;
        }
    }}
void input(int* pn, char* ps, char* pt, char* pd, int num)                    //    输入函数
{
    if (num == 1 || num == 2 || num == 3 || num
```

```
    cout << "1.基本解" << endl;
    cout << "2.基本解(步数记录)" << endl;
    cout << "3.内部数组显示(横向)" << endl;
    cout << "4.内部数组显示(纵向+横向)" << endl;
    cout << "5.图形解-预备-画三个圆柱" << endl;
    cout << "6.图形解-预备-在起始柱上画n个盘子" << endl;
    cout << "7.图形解-预备-第一次移动" << endl;
    cout << "8.图形解-自动移动版本" << endl;
    cout << "9.图形解-游戏版" << endl;
    cout << "0.退出" << endl;
    cout << "--------------------------------" << endl;
    cout << "[请选择:] ";
    while (1)
    {
        char num = _getch();
        switch (num)
        {
            case '0':
                cout << 0;
                return 0;
            case '1':
                cout << 1;
                return 1;
            case '2':
                cout << 2;
                return 2;
            case '3':
                cout << 3;
                return 3;
            case '4':
                cout << 4;
                return 4;
            case '5':
                cout << 5;
                return 5;
            case '6':
                cout << 6;
                return 6;
            case '7':
                cout << 7;
                return 7;
            case '8':
                cout << 8;
                return 8;
            case '9':
                cout << 9;
                return 9;
            default:
                ;
        }
    }}
声明:
int menu();
int end(int);
void input(int*, char*, char*, char*, int);
== 4 || num == 6 || num == 7 || num == 8 || num == 9)
    {
        while (1)
        {
            cout << "请输入汉诺塔的层数(1-10)" << endl;
            cin >> *pn;
            while (cin.fail())
            {
                cin.clear();
                cin.ignore(1024, '\n');
                cout << "请输入汉诺塔的层数(1-10)" << endl;
                cin >> *pn;
            }
            if (*pn >= 1 && *pn <= 10)
                break;
            else
            {
                cin.clear();
                cin.ignore(1024, '\n');
            }
        }
        cin.clear();
        cin.ignore(1024, '\n');
        while (1)
        {
            cout << "请输入起始柱(A-C)" << endl;
            cin >> *ps;
            while (cin.fail())
            {
                cin.clear();
                cin.ignore(1024, '\n');
                cout << "请输入起始柱(A-C)" << endl;
                cin >> *ps;
            }
            if (*ps == 'A' || *ps == 'B' || *ps == 'C' || *ps == 'a' || *ps == 'b' || *ps == 'c')
                break;
            else
            {
                cin.clear();
                cin.ignore(1024, '\n');
            }
        }
        cin.clear();
        cin.ignore(1024, '\n');
        while (1)
        {
            cout << "请输入目标柱(A-C)" << endl;
            cin >> *pd;
            while (cin.fail())
            {
```

装 订 线

```
void hanoi(int n, char src, char tmp, char dst, int num);
void choose(int n, char src, char tmp, char dst, int num);
void first(int n, char src, char dst);
void second(int n, char src, char dst);
void heng(int n, char src, char dst);
void shu(int n, char src, char dst, int num);
void fourth(int n, char src, char dst, int num);
void draw(int n, char src, char dst, int num);
void drawplate(int n, char src, char dst, int num);
void move(int n, char src, char dst, int num);
void eighth(int n, char src, char dst, int num);
void ninth(int n, char src, char tmp, char dst, int num);
void feifa();
```

头文件

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <iomanip>
#include <tchar.h>
#include <string.h>
#include <Windows.h>
#include <conio.h>
#include "cmd_console_tools.h"
#include "hanoi.h"
```

静态全局变量

```
static int x = 1;
static int t = 1;
static int a[10] = {}, b[10] = {}, c[10] = {};
static int top1 = 0, top2 = 0, top3 = 0;
```

```
void hanoi(int n, char src, char tmp, char dst, int num)                    //    递归函数
{
    if (n == 1)
        choose(n, src, tmp, dst, num);
    else
    {
        hanoi(n - 1, src, dst, tmp, num);
        choose(n, src, tmp, dst, num);
        hanoi(n - 1, tmp, src, dst, num);
    }
}
void choose(int n, char src, char tmp, char dst, int num)               //  选择递归函数里应该用哪个函数
{
    if (src == 'A' && dst == 'B')
        b[top2++] = a[--top1];
    else if (src == 'A' && dst == 'C')
        c[top3++] = a[--top1];
    else if (src == 'B' && dst == 'C')
        c[top3++] = a[--top1];
    else if (src == 'B' && dst == 'A')
        a[top1++] = b[--top2];
    else if (src == 'C' && dst == 'A')
        a[top1++] = c[--top3];
    else if (src == 'C' && dst == 'B')
        b[top2++] = c[--top3];
}
void first(int n, char src, char dst)
{
    printf("%c->%c\n", src, dst);
}
void second(int n, char src, char dst)
{
    printf("%c->%c\n", src, dst);
}
void heng(int n, char src, char dst)
{
    printf("%c->%c\n", src, dst);
}
void shu(int n, char src, char dst, int num)
{
    printf("%c->%c\n", src, dst);
}
void fourth(int n, char src, char dst, int num)
{
    printf("%c->%c\n", src, dst);
}
void draw(int n, char src, char dst, int num)
{
    printf("%c->%c\n", src, dst);
}
void drawplate(int n, char src, char dst, int num)
{
    printf("%c->%c\n", src, dst);
}
void move(int n, char src, char dst, int num)
{
    printf("%c->%c\n", src, dst);
}
void eighth(int n, char src, char dst, int num)
{
    printf("%c->%c\n", src, dst);
}
```

```
void ninth(int n, char src, char tmp, char dst, int num)                    //      第九项函数
{
    while (1)
    {
        char whz[25] = {};
        int i = 0;
        while (!(i == 3 && whz[2] == '\r' || i == 2 && (whz[0] == 'q' || whz[0] == 'Q') && whz[1] == '\r'))
        {
            cin.clear();
            cin.ignore(1024, '\n');
            cout << "请输入目标柱(A-C)" << endl;
            cin >> *pd;
        }
        if (*pd == *ps || (*pd - *ps) == 32 || (*pd - *ps) == -32)
        {
            if (*ps > 95)
                *ps = *ps - 32;
            cout << "目标柱(" << *ps << ")不能与起始柱(" << *ps << ")相同" << endl;
            continue;
        }
        if (*pd == 'A' || *pd == 'B' || *pd == 'C' || *pd == 'a' || *pd == 'b' || *pd == 'c')
            break;
        else
        {
            cin.clear();
            cin.ignore(1024, '\n');
        }
    }
    cin.clear();
    cin.ignore(1024, '\n');
    if (*ps > 95)
        *ps = *ps - 32;
    if (*pd > 95)
        *pd = *pd - 32;
    if (*ps == 'A' && *pd == 'B')
        *pt = 'C';
    if (*ps == 'A' && *pd == 'C')
        *pt = 'B';
    if (*ps == 'B' && *pd == 'A')
        *pt = 'C';
    if (*ps == 'B' && *pd == 'C')
        *pt = 'A';
    if (*ps == 'C' && *pd == 'B')
        *pt = 'A';
    if (*ps == 'C' && *pd == 'A')
        *pt = 'B';
}
```

装 订 线

```
            c[top3++] = b[--top2];
        else if (src == 'B' && dst == 'A')
            a[top1++] = b[--top2];
        else if (src == 'C' && dst == 'A')
            a[top1++] = c[--top3];
        else
            b[top2++] = c[--top3];
        if (num == 1)
            first(n, src, dst);
        else if (num == 2)
            second(n, src, dst);
        else if (num == 3)
            heng(n, src, dst);
        else if (num == 4)
            fourth(n, src, dst, num);
        else if (num == 8)
            eighth(n, src, dst, num);
}


void drawplate(int n, char src, char dst, int num)
//    画盘子
{
    int x, y = 14;
    if (src == 'A')
        x = 12;
    else if (src == 'B')
        x = 44;
    else if (src == 'C')
        x = 76;
    int color = n;
    for (int i = 0; i < n; i++)
    {
        Sleep(30);
        cct_showch(x - color, y, ' ', color, 0,
2 * color + 1);
        color--;
        y--;
    }
}
```

```
        {
            if (i == 20)
            {
                cct_showch(60, 34, ' ', 0, 7,
20);
                cct_gotoxy(60, 34);
                i = 0;
            }
            whz[i] = _getche();
            if (whz[i] == ' ')
            {
                cout << '\b';
                i--;
            }
            if (whz[i] == '\b')
            {
                if (i == 0)
                    cout << ' ';
                else
                    cout << whz[i - 1];
                i--;
            }
            i++;
            if (!(i == 3 && whz[2] == '\r' || i
== 2 && (whz[0] == 'q' || whz[0] == 'Q') && whz[1]
== '\r') && whz[i - 1] == '\r')
            {
                cct_showch(60, 34, ' ', 0, 7,
20);
                cct_gotoxy(60, 34);
                i = 0;
            }
        }
        src = whz[0];
        tmp = whz[1];
        if (src == 'q' || src == 'Q')
        {
            cout << endl;
            cout << "游戏中止!!!!!" << endl;
            break;
        }
        if ((src == 'A' || src == 'B' || src ==
'C' || src == 'a' || src == 'b' || src == 'c')
```

装 订 线

```
void draw(int n, char src, char dst, int num)
{
    if (num != 5)
    {
        cct_gotoxy(0, 0);
        cout << "从 " << src << " 移动到 " << dst << ", 共 " << n << " 层";
    }
    cct_setcursor(3);
    cct_showch(1, 15, ' ', 14, 0, 23);
    cct_showch(33, 15, ' ', 14, 0, 23);
    cct_showch(65, 15, ' ', 14, 0, 23);
    for (int y = 14; y >= 3; y--)
    {
        Sleep(50);
        cct_showch(12, y, ' ', 14, 0, 1);
        cct_showch(44, y, ' ', 14, 0, 1);
        cct_showch(76, y, ' ', 14, 0, 1);
    }
    if (num == 6 || num == 7 || num == 8 || num == 9)
        drawplate(n, src, dst, num);
    if (num == 7)
    {
        if (src == 'A' && dst == 'B')
            b[top2++] = a[--top1];
        else if (src == 'A' && dst == 'C')
            c[top3++] = a[--top1];
        else if (src == 'B' && dst == 'C')
            c[top3++] = b[--top2];
        else if (src == 'B' && dst == 'A')
            a[top1++] = b[--top2];
        else if (src == 'C' && dst == 'A')
            a[top1++] = c[--top3];
        else
            b[top2++] = c[--top3];
    }
    cct_setcolor(0, 7);
    cct_gotoxy(0, 37);
    if (num == 9)
    {
        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) : ";
        cct_setcursor(2);
    }
}
```

```
&& (tmp == 'A' || tmp == 'B' || tmp == 'C' || tmp == 'a' || tmp == 'b' || tmp == 'c'))
        {
            if (tmp != src && (tmp - src) != 32 && (tmp - src) != -32)
            {
                if (src > 95)
                    src = src - 32;
                if (tmp > 95)
                    tmp = tmp - 32;
                if (src == 'A' && top1 == 0 || src == 'B' && top2 == 0 || src == 'C' && top3 == 0)
                {
                    cout << endl;
                    cout << "源柱为空!";
                    Sleep(1000);
                    cct_showch(0, 35, ' ', 0, 7, 30);
                    cct_showch(60, 34, ' ', 0, 7, 30);
                    cct_gotoxy(60, 34);
                    continue;
                }
                else
                {
                    if (src == 'A' && tmp == 'B')
                    {
                        if (b[top2 - 1] != 0 && a[top1 - 1] > b[top2 - 1])
                        {
                            feifa();
                            continue;
                        }
                    }
```

装 订 线

装 订 线

```
void first(int n, char src, char dst)
{
    cout << setw(2) << n << "#" << " " << src << "---->" << dst << endl;
}
void second(int n, char src, char dst)
{
    cout << "第" << setw(4) << x << " 步(" << setw(2) << n << "#: " << src << "-->" << dst << ")" << endl;
    x++;
}
void heng(int n, char src, char dst)
{
    cout << "第" << setw(4) << x << " 步(" << setw(2) << n << "#: " << src << "-->" << dst << ")" << "A:";
    for (int i = 0; i < top1; i++)
        cout << setw(2) << a[i];
    cout << setfill(' ') << setw((10 - top1) * 2 + 1) << ' ' << "B:";
    for (int i = 0; i < top2; i++)
        cout << setw(2) << b[i];
    cout << setfill(' ') << setw((10 - top2) * 2 + 1) << ' ' << "C:";
    for (int i = 0; i < top3; i++)
        cout << setw(2) << c[i];
    cout << setfill(' ') << setw((10 - top3) * 2 + 1) << ' ' << endl;
    x++;
}
void fourth(int n, char src, char dst, int num)
//   第四项   函数
{
    cct_gotoxy(0, 17);
    heng(n, src, dst);
    shu(n, src, dst, num);
    if (t == 0)
        while (_getch() != '\r');
    else
        Sleep(t);
}
void eighth(int n, char src, char dst, int num)
//   第八项    函数
{   cct_setcolor(0, 7);
    cct_gotoxy(0, 32);
    heng(n, src, dst);
    shu(n, src, dst, num);
    move(n, src, dst, num);
    if (t == 0)
        while (_getch() != '\r')
            ;
    else
        Sleep(t);}
```

```
void move(int n, char src, char dst, int num)
//   移动色块   公用一个函数
{
    if (src == 'A' && dst == 'B')
    {
        int x = 12, y;
        for (y = 14 - top1; y > 1; y--)
        {
            cct_showch(x - a[top1], y, ' ', 0, 0, a[top1]);
            if (y < 3)
                cct_showch(x, y, ' ', 0, 0, 1);
            else
                cct_showch(x, y, ' ', 14, 0, 1);
            cct_showch(x + 1, y, ' ', 0, 0, a[top1]);
            cct_showch(x - a[top1], y - 1, ' ', a[top1], 0, 2 * a[top1] + 1);
            if (t != 0)
                Sleep(t);
            else
                Sleep(30);
        }
        for (x = x - a[top1]; x < 44 - a[top1]; x++)
        {
            cct_showch(x, y, ' ', 0, 0, 2 * a[top1] + 1);
            cct_showch(x + 1, y, ' ', a[top1], 0, 2 * a[top1] + 1);
            if (t != 0)
                Sleep(t);
            else
                Sleep(30);
        }
        for (y = 1; y <= 14 - top2; y++)
        {
            cct_showch(x, y, ' ', 0, 0, a[top1]);
            if (y < 3)
                cct_showch(x + a[top1], y, ' ', 0, 0, 1);
            else
                cct_showch(x + a[top1], y, ' ', 14, 0, 1);
            cct_showch(x + a[top1] + 1, y, ' ', 0, 0, a[top1]);
            cct_showch(x, y + 1, ' ', a[top1], 0, 2 * a[top1] + 1);
            if (t != 0)
                Sleep(t);
            Else Sleep(30);}}
```