同济大学

高程 综合题一

汉诺塔 实验报告

学　　号 2152118
姓　　名 史君宝
班　　级 计科一班

完成日期 2022.11.27

1. 汉诺塔综合实验

1.1. 题目描述

本次的大作业我们主要将原来的所有的汉诺塔的小作业合在一起，共同完成本次汉诺塔的大作业。下面是题目描述：

（1）完成汉诺塔的基本解。

（2）完成汉诺塔的步数计算，并给出结果。

（3）完成汉诺塔的数组形式，并将数组的内容横向展示出来。

（4）完成汉诺塔的数组形式，同时打印出图像，在图像上完成移动

（5）用已经给的辅助函数画出汉诺塔柱子。

（6）用已经给的辅助函数画出汉诺塔柱，并画出汉诺塔的圆盘。

（7）在画出的汉诺塔的基础下，完成汉诺塔的第一次移动。

（8）在画出的汉诺塔的基础上，根据延时提示完成自动移动的操作。

（9）设计汉诺塔游戏，在画出的汉诺塔的基础上，可以通过输入控制圆盘的移动，并给出游戏过程中的提示。

（10）退出程序。

2. 整体设计思路

2.1. 整体设计思路

本人在做这一题的时候格外注重程序的复用性，旨在用少量的函数完成程序，并减少相关的重复的部分。主要的设计思路是先确定某个函数要完成的任务，然后在完成任务的目标上开始编写函数。在各函数完成之后，通过一些操作充当胶水的作用，将所用的函数拼接起来，并最终完成程序设计。

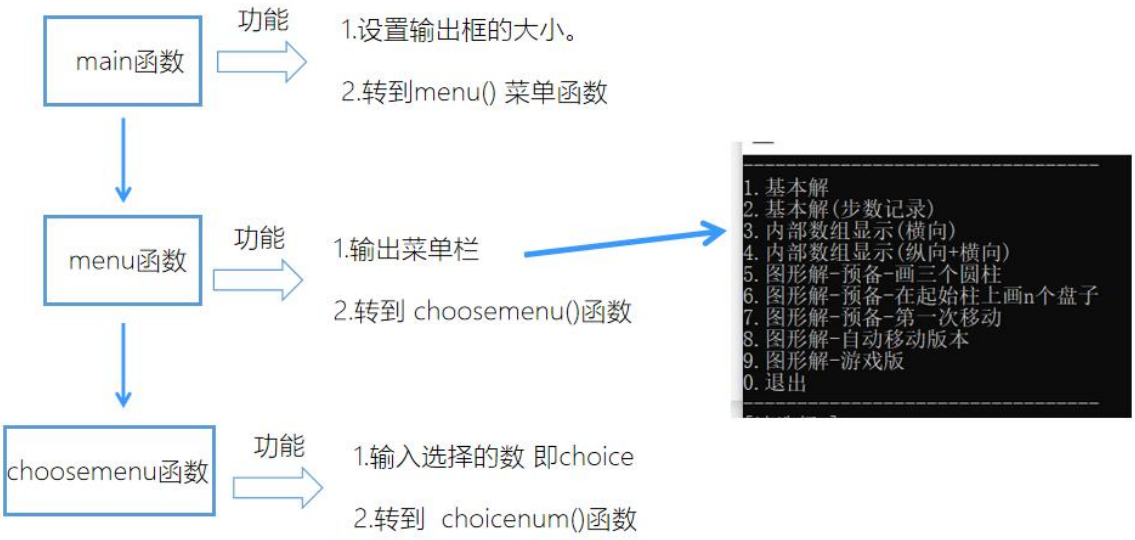
因此在程序中基本不会出现如果一个功能在两个操作中略有不同，会单独地为这两个程序分别设计一个函数的情况，主要通过一系列的选择结构来实现情况的分流。

但是过程中出现了不顺利，由于沿用上次作业中的一维数组作为储存圆盘的数据方式，这就导致了部分代码不能同时表示三个柱子，比如在确定柱子的名称并对对应的数组进行操作的时候就十分困难，有部分代码难以重复利用。

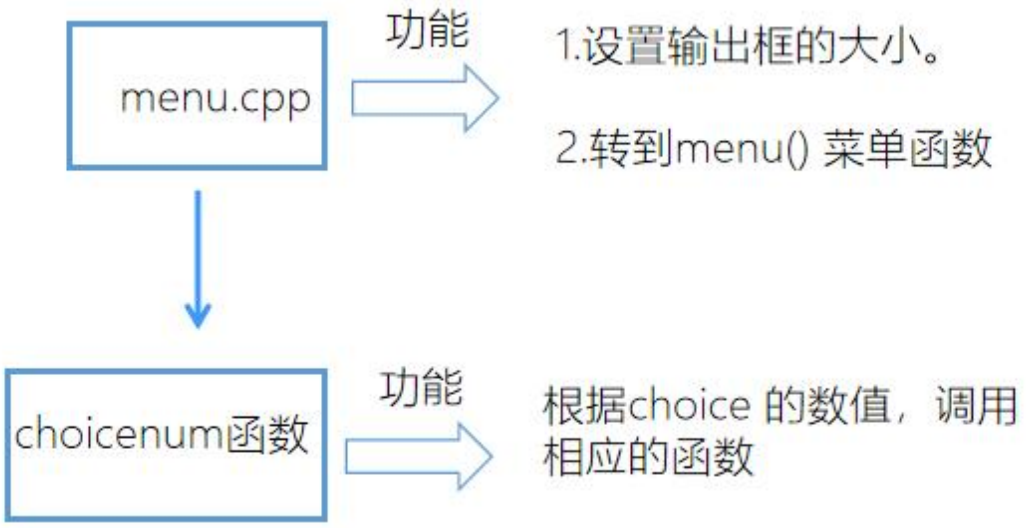
3. 主要功能的实现

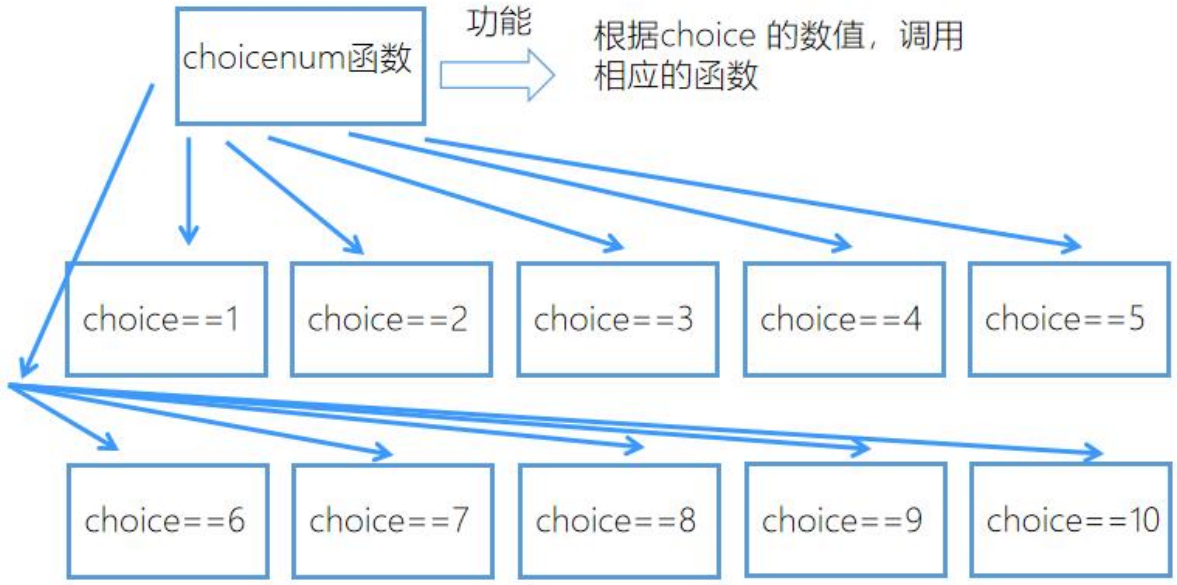
3.1.main.cpp 中的函数 （main函数）

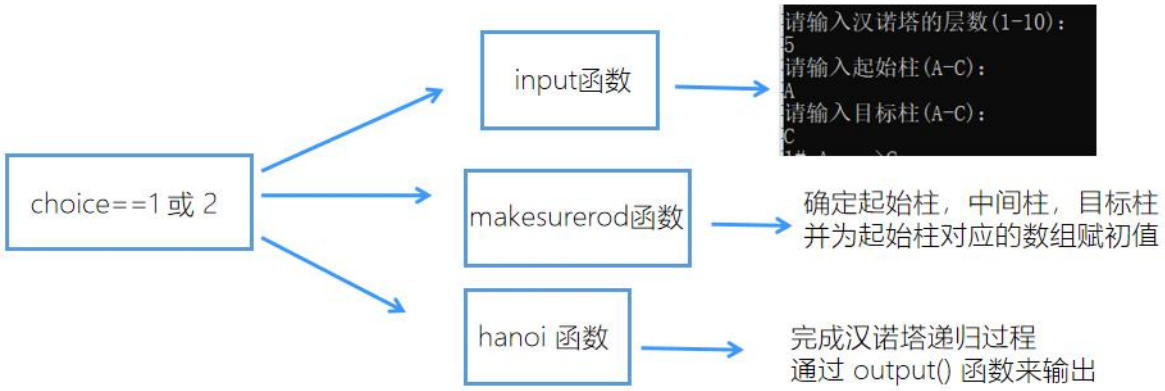
3.2.menu.cpp 中的函数 （menu 函数 和 choosemenu 函数）

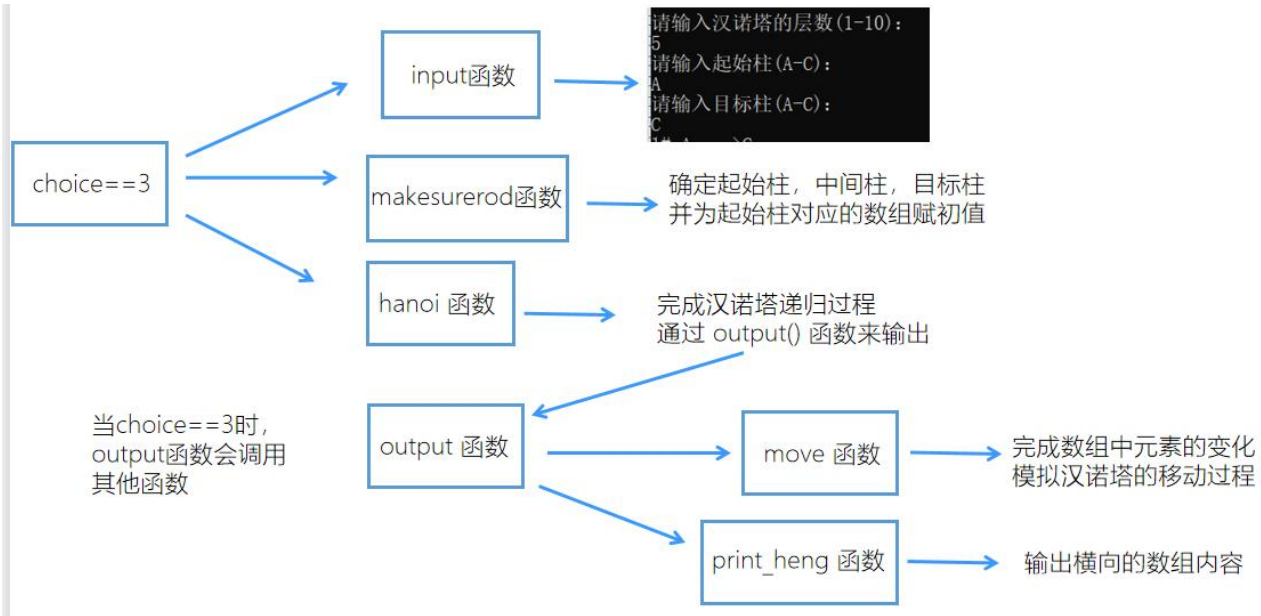


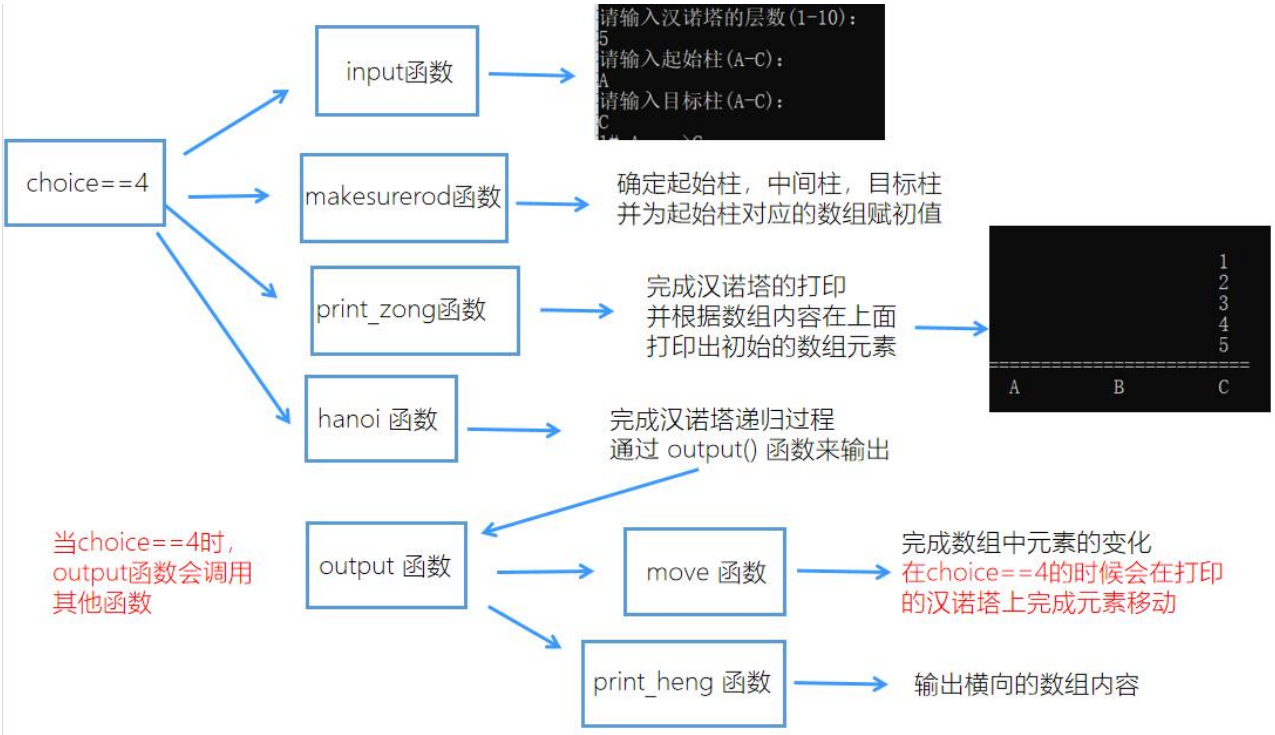
3.3.multiple_solutions.cpp 中的函数



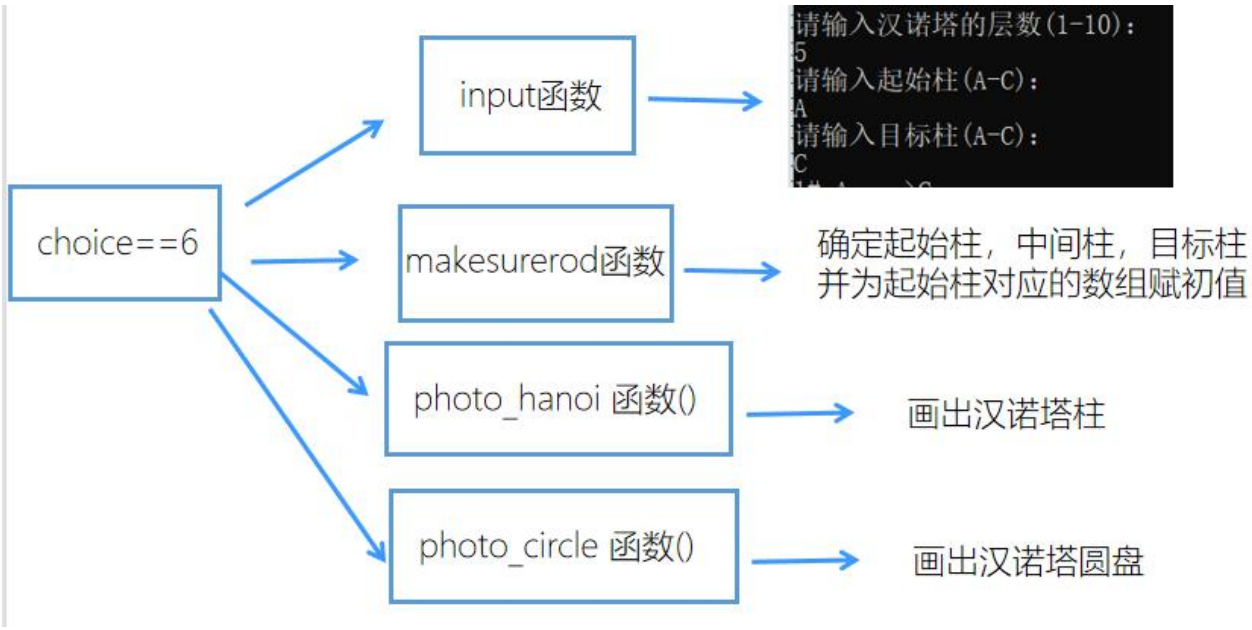

choicenum函数
功能
根据choice 的数值，调用
相应的函数
choice==1
choice==2
choice==3
choice==4
choice==5
choice==6
choice==7
choice==8
choice==9
choice==10

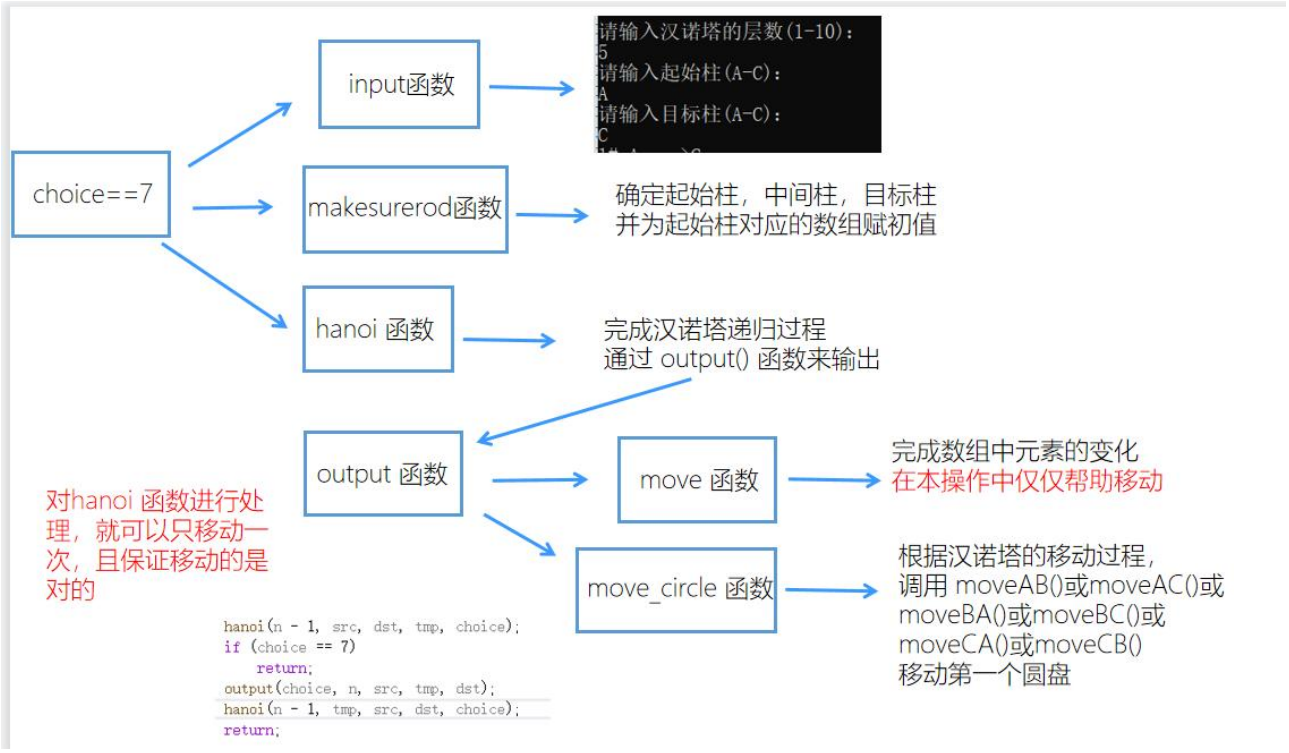

choice==1 或 2
input函数
请输入汉诺塔的层数(1-10)：
5
请输入起始柱(A-C)：
A
请输入目标柱(A-C)：
C
makesurerod函数
确定起始柱，中间柱，目标柱
并为起始柱对应的数组赋初值
hanoi 函数
完成汉诺塔递归过程
通过 output() 函数来输出

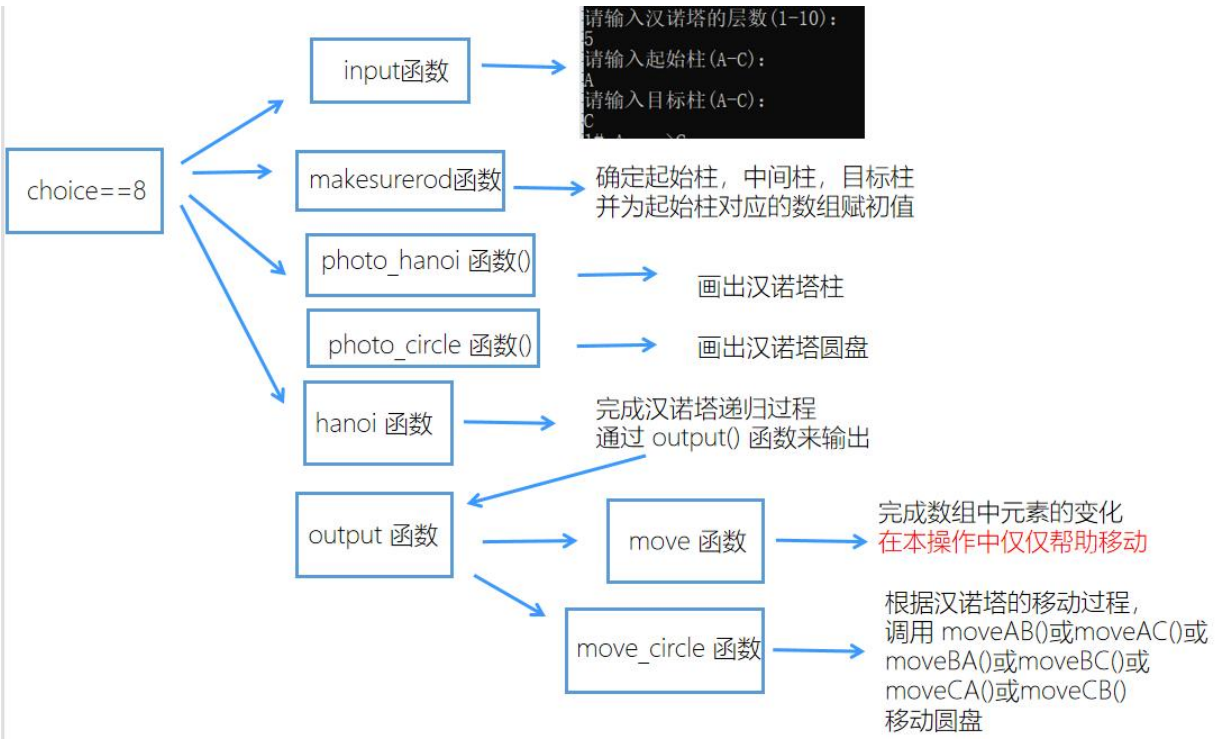

choice==3
input函数
请输入汉诺塔的层数(1-10)：
5
请输入起始柱(A-C)：
A
请输入目标柱(A-C)：
C
makesurerod函数
确定起始柱，中间柱，目标柱
并为起始柱对应的数组赋初值
hanoi 函数
完成汉诺塔递归过程
通过 output() 函数来输出
当choice==3时，
output函数会调用
其他函数
output 函数
move 函数
完成数组中元素的变化
模拟汉诺塔的移动过程
print_heng 函数
输出横向的数组内容

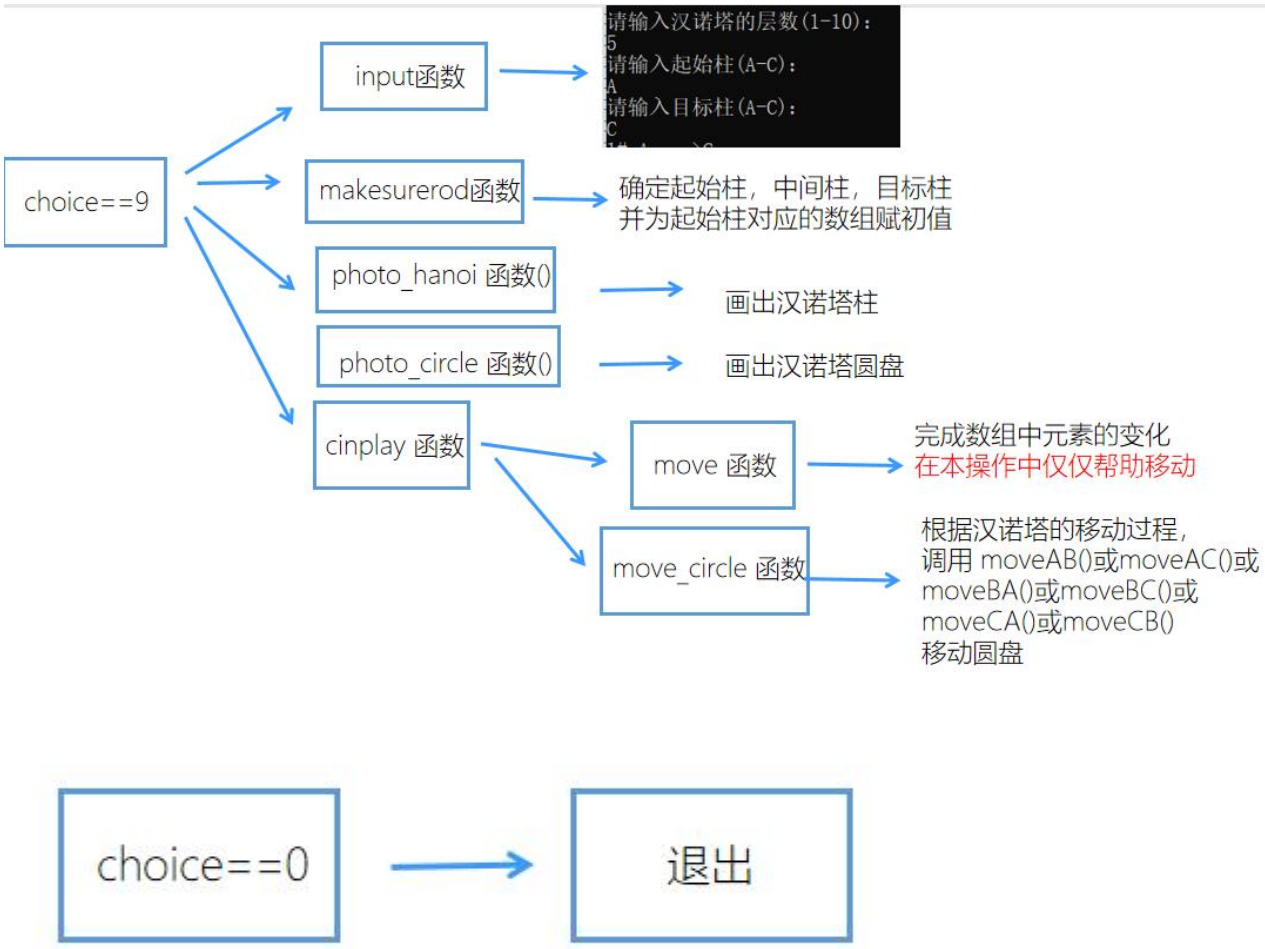

choice==4
input函数
请输入汉诺塔的层数(1-10)：
5
请输入起始柱(A-C)：
A
请输入目标柱(A-C)：
C
makesurerod函数
确定起始柱，中间柱，目标柱
并为起始柱对应的数组赋初值
print_zong函数
完成汉诺塔的打印
并根据数组内容在上面
打印出初始的数组元素
1
2
3
4
5
A B C
hanoi 函数
完成汉诺塔递归过程
通过 output() 函数来输出
当choice==4时，
output函数会调用
其他函数
output 函数
move 函数
完成数组中元素的变化
在choice==4的时候会在打印
的汉诺塔上完成元素移动
print_heng 函数
输出横向的数组内容


choice==5
photo_hanoi 函数()
画出汉诺塔柱


choice==6
input函数
请输入汉诺塔的层数(1-10)：
5
请输入起始柱(A-C)：
A
请输入目标柱(A-C)：
C
makesurerod函数
确定起始柱，中间柱，目标柱
并为起始柱对应的数组赋初值
photo_hanoi 函数()
画出汉诺塔柱
photo_circle 函数()
画出汉诺塔圆盘


choice==7
input函数
请输入汉诺塔的层数(1-10):
5
请输入起始柱(A-C):
A
请输入目标柱(A-C):
C
makesurerod函数
确定起始柱，中间柱，目标柱
并为起始柱对应的数组赋初值
hanoi 函数
完成汉诺塔递归过程
通过 output() 函数来输出
output 函数
move 函数
完成数组中元素的变化
在本操作中仅仅帮助移动
move_circle 函数
根据汉诺塔的移动过程，
调用 moveAB()或moveAC()或
moveBA()或moveBC()或
moveCA()或moveCB()
移动第一个圆盘
对hanoi 函数进行处理，就可以只移动一次，且保证移动的是对的
hanoi(n - 1, src, dst, tmp, choice);
if (choice == 7)
return;
output(choice, n, src, tmp, dst);
hanoi(n - 1, tmp, src, dst, choice);
return;

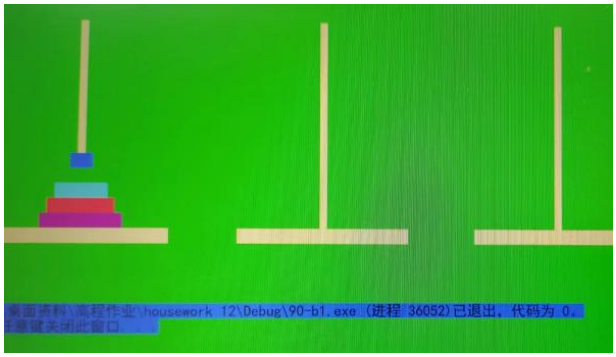

choice==8
input函数
请输入汉诺塔的层数(1-10):
5
请输入起始柱(A-C):
A
请输入目标柱(A-C):
C
makesurerod函数
确定起始柱，中间柱，目标柱
并为起始柱对应的数组赋初值
photo_hanoi 函数()
画出汉诺塔柱
photo_circle 函数()
画出汉诺塔圆盘
hanoi 函数
完成汉诺塔递归过程
通过 output() 函数来输出
output 函数
move 函数
完成数组中元素的变化
在本操作中仅仅帮助移动
move_circle 函数
根据汉诺塔的移动过程，
调用 moveAB()或moveAC()或
moveBA()或moveBC()或
moveCA()或moveCB()
移动圆盘



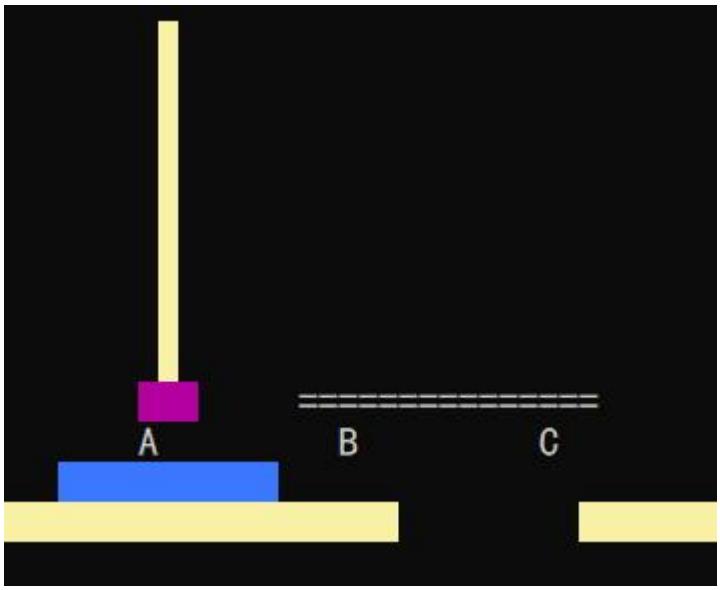
4. 调试过程碰到的问题

4.1. 调试中出现的问题

（1）程序过程中不太理解颜色的设置的问题，常常会出现颜色的设置问题。比如在设置颜色之后，忘了把颜色调成黑色，后导致下面的问题。



（2）在程序过程中需要一定的方式，将程序的输出转到一定的位置输出，如果不能很好的控制就会导致一定的错误。

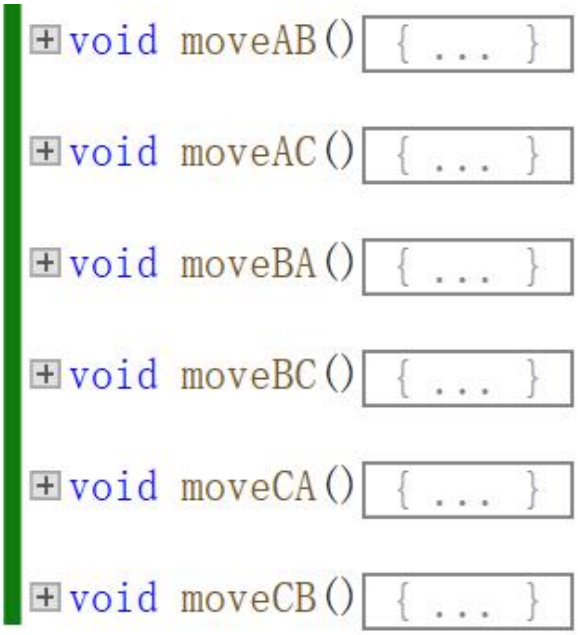


（3）在程序过程中在设置延时效果为0的时候，在choice==8的时候，本来应该是按一次回车，完成一次圆盘的移动的，但是由于不当的操作，将wait()函数放在不对的位置处，变成了按一次操作仅仅移动一小格。

（4）在程序过程中一个格外需要注重的问题，就是数据结构的选择不太合适。在之前的作业中我采用的是三个一维数组的方式来储存圆盘，但是在这一题中就不太好了。由于柱子被分为三个变量，彼此之间除了名字类似，很难进行统一化的操作，就造成了大量的代码相同，但是难以去掉。举个例子：

```
//确定起始柱是哪个柱，并为相对应的数组赋值
if (qishi == 'A')
{
    a_top = n;
    for (int i = 0; i < n; ++i)
        a[i] = n - i;
}
else if (qishi == 'B')
{
    b_top = n;
    for (int i = 0; i < n; ++i)
        b[i] = n - i;
}
else
{
    c_top = n;
    for (int i = 0; i < n; ++i)
        c[i] = n - i;
}
```

仅仅是为柱子对应的数组赋初值，由于三个一维数组彼此之间相互独立，缺少必要的联系，就导致了大量代码堆积，三个if语句完成的是同一个操作，但是却难以将代码进行删减。在之后一旦输出新的值，在判断柱子的时候就会出现多次判断的情况，比如，在图形上移动圆盘的时候就用了六个函数，大大增加了维护的困难。



5. 心得体会

5.1. 心得体会

（1）在本题中，最大的问题就是数据结构的选择的不太适合。就像前面所说的，由于数据结构的不适宜，导致了代码的大量的堆积，还难以消除，这就在一定程度上增加了维护的困难性。

在之后的学习中，一定要格外注重这个问题，一种好的数据结构，或者数据存储方式，能够极大程度上简化自己的代码。

（2）同时，在程序设计的时候一定要处理好各个函数之间的关系，尽量做到函数自身的独立性，同时保证调用函数的时候思路清晰明确。

（3）在做大作业的时候，一定要先仔细研究程序的细节问题，并做好笔记，构建一定的程序框图，这样可以简化程序设计的过程，帮助解决问题。

5.2. 问题1

在做一些复杂的程序的时候，应该将程序分为若干小题，在这种不断提升的过程中，能够帮助养成

好的编程习惯和思考方式。

5.3. 问题2

在做题的过程中，我很多程序都是利用之前的程序直接拿来用了，在很大程度实现了代码的复用性，但是也造成了一定的问题。建议自己下一次不是拿来一点不改，进行一点点修改，能够方便我们的处理。

5.4. 问题3

在程序设计的时候一定要处理好各个函数之间的关系，尽量做到函数自身的独立性，同时保证调用函数的时候思路清晰明确。

我在本题中尽量做到的是用少量的函数完成程序，并减少相关的重复的部分。主要的设计思路是先确定某个函数要完成的任务，然后在完成任务的目标上开始编写函数。在各函数完成之后，通过一些操作充当胶水的作用，将所用的函数拼接起来，并最终完成程序设计。

我认为这种方式是适合我的方式。

6. 附件：源程序

```
/* 计科 2152118 史君宝 */

#include <iostream>
#include <iomanip>
#include <Windows.h>
#include "cmd_console_tools.h"
#include "hanoi.h"
using namespace std;

static int a_top = 0, b_top = 0, c_top = 0;
static int cishu = 0;
static int a[10], b[10], c[10];
static int speed=4;

void menu();
void return_menu(int choice)
{
    char m;
    if (choice == 1 || choice == 2||choice==3)
        cout << endl << endl;
    else
        cct_gotoxy(0, 35);
    cout << "按回车键继续";
```

```
    while ((m = getchar()) != '\n');
    a_top = b_top = c_top = cishu = 0;
    speed = 4;
    for (int i = 0; i < 10; ++i)
        a[i] = b[i] = c[i] = 0;
    cct_cls();
    menu();
}

void wait();                //设置等待方式的函数


void moveAB()
{
    int x, y;
    for (y = 14 - a_top; y >= 1; --y) {
        /* 在坐标(x,2)位置处连续打印10个字符 */
        cct_showch(12 - a[a_top], y, ' ', 14 - a[a_top], 0, 2 * a[a_top] + 1);

        if (speed != 0)
            wait();
        else
            Sleep(100);

        if (y > 1) {
            /* 清除显示(最后一次保留)，清除方法为用正常颜色+空格重画一遍刚才的位置 */
            cct_showch(12 - a[a_top], y, ' ', 0, 0, 2 * a[a_top]);//黑色
            if (y >= 3)
                cct_showch(12, y, ' ', 14, 0, 1);//黑色
            else
                cct_showch(12, y, ' ', 0, 0, 1);//黑色
            cct_showch(13, y, ' ', 0, 0, 2 * a[a_top]);//黑色
        }
    } //end of for
}

void moveAC();   同上

void moveBA();

void moveBC();

void moveCA();

void moveCB();

//画出汉诺塔的柱子
void photo_hanoi()
```

```
{
    cct_showch(1, 15, ' ', 14, 0, 23);
    cct_showch(33, 15, ' ', 14, 0, 23);
    cct_showch(65, 15, ' ', 14, 0, 23);
    for (int i = 15; i >= 3; --i)
    {
        Sleep(100);
        cct_showch(12, i, ' ', 14, 0, 1);
        cct_showch(44, i, ' ', 14, 0, 1);
        cct_showch(76, i, ' ', 14, 0, 1);
    }
    cct_gotoxy(0, 20);
    cct_setcolor(0, 7);
}

//画出汉诺塔的圆盘
void photo_circle(char qishi)
{
    if (qishi == 'A')
        for (int i = 0; i < a_top; ++i)
        {
            Sleep(1000);
            for (int j = 12 - a_top + i; j <= 12 + a_top - i; ++j)
                cct_showch(j, 14 - i, ' ', 14 - a[i], 0, 1);
        }
    if (qishi == 'B')
       ……
    if (qishi == 'C')
        ……
    cct_setcolor(0, 7);
}

//该操作用于实现圆盘的移动，并实现对应情况的分流
void move_circle(int choice,char qishi,char zhongjian,char mubiao) ;

//打印横向的数组
void print_heng(int n,char qishi,char zhongjina,char mubiao,int choice)
{
    if(choice==3)
        printf("第%4d 步(%2d#: %c-->%c)", ++cishu, n, qishi, mubiao);
    else
        cout << "第" << setw(4) << ++cishu << "步(" << n << "# " << qishi << "--> " << mubiao
<< ")";

    cout << " A:";
    for (int i = 0; i < a_top; ++i)
        printf("%2d", a[i]);
}
```

```
//移动过程中的数组变化
void move(char qishi, char zhongjian, char mubiao, int choice, int X)
{
    if (qishi == 'A')
    {
        if (mubiao == 'B')
        {
            if (choice == 9 && (a[a_top - 1] > b[b_top -1] && b[b_top - 1] != 0))
            {
                cct_gotoxy(0,31);
                cout << "大盘压小盘，非法移动!" << endl;
                Sleep(1000);
                cct_gotoxy(0, 31);
                cout << "                                                    ";
                return;
            }
            if (choice == 9 && a_top == 0)
            {
                cct_gotoxy(0, 31);
                cout << "源柱为空!" << endl;
                Sleep(1000);
                cct_gotoxy(0, 31);
                cout << "                                                    ";
                return;
            }
            --a_top;
            b[b_top] = a[a_top];
            if (choice == 4 || choice == 8 || choice == 9)
            {
                cct_gotoxy(11, 11 - a_top + X);
                cout << " ";
                cct_gotoxy(21, 11 - b_top + X);
                cout << b[b_top];
            }
            ++b_top;
            if (choice == 9)
            {
                moveAB();
                cct_setcolor(0, 7);
                cct_gotoxy(0, 28);
                print_heng(b[b_top - 1], qishi, zhongjian, mubiao, 9);
            }
        }

}

//打印纵向的数组
```

```
void print_zong(char qishi,int X);

//输入函数汇总(根据不同的choice采用不同的输入)
void input(int &n,char &qishi,char &mubiao,int &choice);

//输出格式的汇总(根据choice的不同采用不同的输出)
void output(int choice,int n, char src, char tmp, char dst);


void cinplay(char qishi,char zhongjian,char mubiao,int n)
{
    char one, two;
    while (1)
    {
        cct_gotoxy(60, 30);
        cin >> one;
        if (one == 'Q'||one=='q')
        {
        }
        cin >> two;
        cin.clear();
        cin.ignore(65535, '\n');
        if (((one >= 'A' && one <= 'C') || (one - 32 >= 'A' && one - 32 <= 'C')) &&
            ((two >= 'A' && two <= 'C') || (two - 32 >= 'A' && two - 32 <= 'C')))
            if (!(one - two == 0 || fabs(one - two) == 32))
                move(one, ' ', two, 9, 15);

            cct_gotoxy(60, 30);
            cout << "              "<<endl<<"           ";
            if ((mubiao == 'A'&&a_top==n)|| (mubiao == 'B' && b_top == n)
                || (mubiao == 'C' && c_top == n))
            {
            }
    }
}

//汉诺塔的递归函数
void hanoi(int n, char src, char tmp, char dst, int choice);

//确定中间柱，并通过引用参数传回中间柱
void makesurerod(int n,char qishi,char mubiao,char &zhongjian)

void choicenum(int choice);
```