高级语言程序设计

汉诺塔综合演示

实验报告

信 16 2352495 张竹和

完成时间：2024.5.16

1. 题目

1.1.菜单页面要求

将之前做的所有汉诺塔的各小题集成在一个程序中，用菜单形式进行选择，即制作一个伪图形界面；

1.2.菜单选项要求

菜单1要求：输出汉诺塔的基本解；
菜单2要求：输出汉诺塔的基本解，并包含步数记录；
菜单3要求：横向输出汉诺塔的每步数组显示；
菜单4要求：纵向输出汉诺塔的每步数组显示；
菜单5要求：在屏幕上画出三根圆柱，需要加延时；
菜单6要求：根据圆盘编号与数量的输入画出初始状态的圆柱和柱上的圆盘，每个圆盘的颜色各不相同，需要加延时；
菜单7要求：在6基础上完成第一个圆盘移动动画，要求必须先上移，再平移，再下移动；
菜单8要求：完成汉诺塔移动过程的完整动画演示；
菜单9要求：汉诺塔游戏，根据人工输入进行动画演示移动，需要检查是否符合汉诺塔的规则，每次合理的移动均需要记录步数，若不符合规则或是全部移动到结束柱均需有相应的输出提示；

2. 整体设计思路

2.1.总体思路

将各个功能分在各个小的自定义函数中，再通过自定义函数进行集成并实现与用户的交互。

2.2.具体论述

程序的核心函数主要包括hanoi_menu(菜单函数，管理用户的输入)、hanoi(汉诺塔完整步骤的递归函数)。除去个别需要单独处理的菜单，主要围绕这两个函数进行补充；

程序的具体处理函数包括包括draw_pillar(根据输入在指定位置画三根柱子)、draw_plate(在指定位置，以指定颜色、指定宽度画盘子)、time_delay(根据输入选择延时时间长短)、vertical_output(根据数组完成竖排的数字输出)、horizontal_out(完成横排的数字输出)、update_abc(根据每步的移动情况进行数组更新)、output_number(每个圆柱的现有圆盘编号的输出)、output_line(每行字母的整体输出)、move_xy(根据坐标进行上下左右移动的函数)、input(汉诺塔输入的函数)；

集成部分选项的函数包括menu_choice(1、2、3、4、7、8菜单项的函数调用)、total_hanoi(3、4、7、8菜单项的函数调用)、move(管理一个盘子从一个柱到另一个柱的整体移动函数调用)；

另外，由于部分菜单的特殊性，单独写了output_9(9的整体输出，除动画)、input_9(9的移动指令

输入)；

通过以上函数的相互调用等完成整体程序的实现。

3. 主要功能的实现

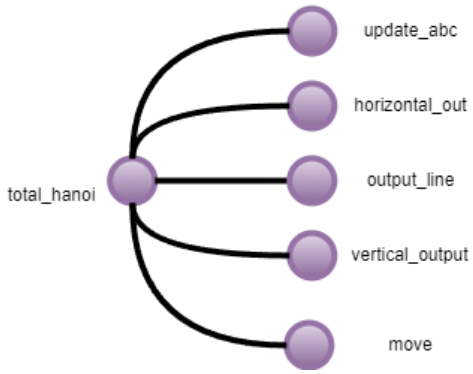
3.1. 自定义函数

为了更清楚的展现，首先将程序中出现过的所有自定义函数(除main)制表如下：

函数	名称	功能
核心函数	hanoi_menu	菜单函数，管理用户的输入
	hanoi	汉诺塔完整步骤的递归函数
集成部分选项的函数	menu_choice	1、2、3、4、7、8菜单项的函数调用
	total_hanoi	3、4、7、8菜单项的函数调用
	move	管理一个盘子从一个柱到另一个柱的整体移动函数调用
具体需求的处理函数	draw_pillar	根据输入在指定位置画三根柱子
	draw_plate	在指定位置，以指定颜色、指定宽度画盘子
	time_delay	根据输入选择延时时间长短
	vertical_output	根据数组完成竖排的数字输出
	horizontal_out	完成横排的数字输出
	update_abc	根据每步的移动情况进行数组更新
	output_number	每个圆柱的现有圆盘编号的输出
	output_line	每行字母的整体输出
	move_xy	根据坐标进行上下左右移动的函数
	input	汉诺塔输入的函数
特定菜单的函数	output_9	9的整体输出，除动画
	input_9	9的移动指令输入

3.2. 主要集成函数

集成部分函数的示意图如下

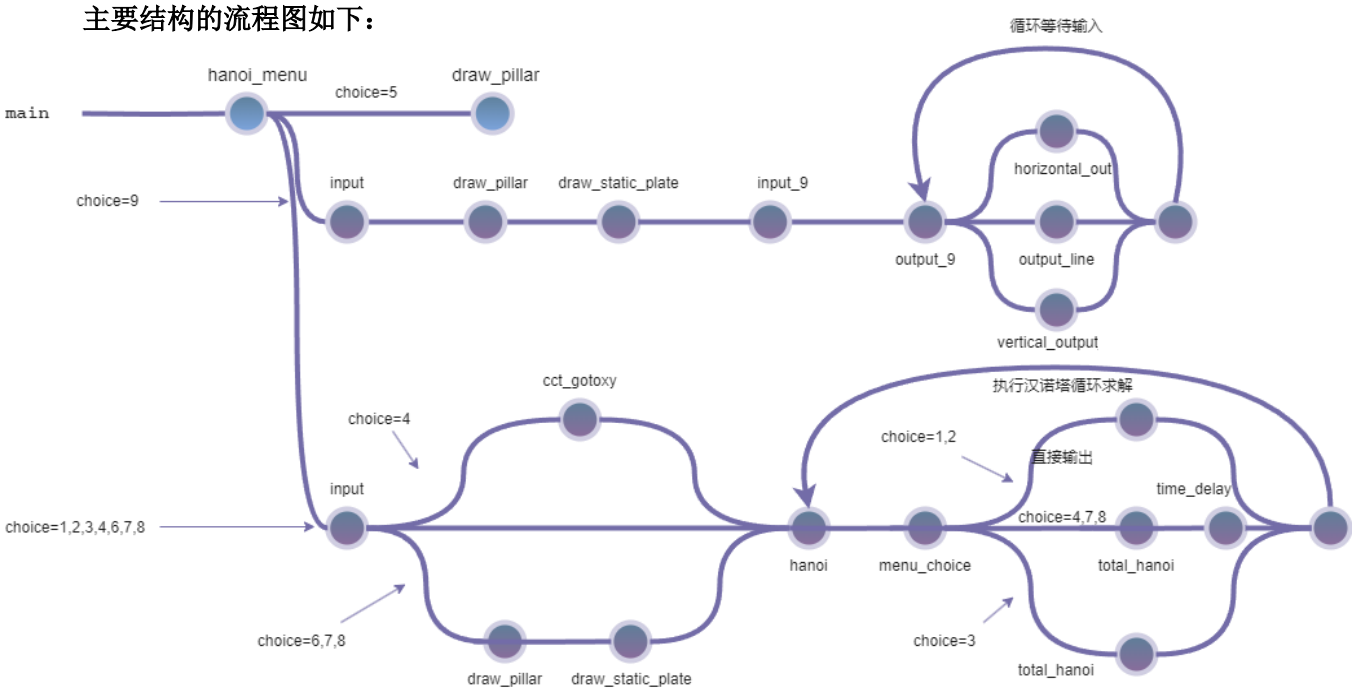


该函数根据菜单项choice的不同在这些函数中选择相应的并依次执行。

3.3. 主要结构

程序开始，首先运行hanoi_manu，等待用户输入菜单选择，根据菜单选择是否进入汉诺塔的输入，若进入，再进行原始图形输出，然后根据菜单选择等待移动的命令输入；或是运行汉诺塔的递归函数，并根据菜单选项选择汉诺塔每步递归结果的输出方式，通过调用相应的不同函数实现输出。

主要结构的流程图如下：



4. 调试过程碰到的问题

问题1：在7、8、9菜单项中，数组更新时，出现混乱(屏幕上出现的盘子大小与颜色均与输入不符)，即数组中对应的数字与预期不同；

解决方案：使用F9键在输出盘子的代码部分设置断点，并用F5进行运行调试，在断点处查看数组中值的大小与预期的差异，并继续使用F9结合F5在数组更新或输入的部分中止，改用F11逐步运行调试，每步检查数组变化，找出导致数组未正确读取或更新的代码。

问题2：同时需要动画演示与横竖排输出时，屏幕输出出现圆柱一整列为黑色、且竖排数字未输出的情况；

解决方案：首先在输出竖排数字的函数处设置断点，F5运行调试，在断点处改用F11逐步运行调试，确定了该输出竖排数字的函数得到了执行，故猜想是坐标的重合问题导致输出竖排数字的位置与柱子黑色的那一列重合，再通过F9+F5+F11进行调试与检查，发现了坐标的问题，并在代码中完成修改。

问题3：盘子移动的函数没有按预期执行；

解决方案：考虑到这是一个相对独立的函数，故另建一项目单独进行该函数的实验，在该项目中使

用F9+F5+F11进行调试修改，直到该函数的输入与输出符合要求，在本问题中即表现为输入初始柱与终止柱，程序通过检索数组得到该柱最上层的盘子的相关参数并实现先上移，再平移，最后下移的整体动画效果。

5. 心得体会

5.1.完成本次作业的心得体会与经验教训

5.1.1. 更多使用自定义函数

多自定义函数，少在函数里写大段的语句来实现功能。除本次作业硬性要求外，如根据每一步的目标柱到终止柱的移动更新数组的函数，又如根据初始坐标和终止坐标来实现盘子移动的函数，以及在指定位置以指定颜色画盘子的函数。通过自定义函数，一方面可以在单独的项目中进行该函数的实验与调试，相比在整个复杂程序中整体调试效率更高；另一方面通过在不同的其他函数中调用该函数，也能提高代码的可读性并方便代码的维护与调试。

5.1.2. 自定义函数使用更多形参

自定义函数中多使用形参，或者使用重载函数。这样能方便函数的使用，在调用函数时更清晰。两种方法各有优劣，使用更多的形参可能需要更多时间对程序进行调试，好处则是代码量会相对少；而使用重载函数在调试上比较简单不需要很多时间，但会导致整体项目的代码量增加，另外也可能对后续修改与维护造成困难。

5.1.3. 熟练使用VS中的debug工具

debug时灵活使用F5、F9、F11以及其他VS的debug工具可以大量节省时间，在对于循环的debug过程中，可以在程序调试中设置断点，这样可以直接从bug出现的那一次循环开始，操作可以为：在相应位置添加需要读取输入的代码(若无)，然后在程序等待输入的时候就可以在需要的位置设置断点，并使用F11进行逐步查验。

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更好？

分为若干小题更好。

分为若干小题逐步完成思路更清晰，每个单独的代码量小不易出错，相对debug的难度小，后续的修改与维护也更容易，如在本题中细分数组的更新函数、动画中画盘子的函数、画柱子的函数、移动盘子的函数和整体移动的函数等待，在单独的项目中写小函数直接放入主程序，查错更方便；

而以一道大题的形式往往思路混乱，体现在代码中导致代码冗长且不够清晰，无论是调试还是后续修改都会有很大麻烦。

5.3.总结在完成汉诺塔若干小题过程中是否考虑了前后小题的关联关系，是否能做到后面小题有效利用前面小题已完成的代码，如何才能更好的重用代码？

在本次作业中，我考虑到前后小题的关联，所以很多后面小题都是在前面小题的代码的基础上直接修改完成的，例如直接修改其中某一个自定义函数中的某些部分，但这样导致在最后大作业的函数整合过程中遇到了麻烦，很多地方需要仔细阅读之前的源程序然后修改整合，效率低下。若想更好地重用代码，我认为需要改进的地方在于，不应该直接修改前面小题源程序的代码，以某一个函数举例，应该在前一题该函数的基础上进行添加，如添加一个形参等待，以此方式实现新的小题，原则应当是“只加不删”。

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？

5.4.1. 自定义函数

对于函数本身而言，在定义过程中尽量考虑其普适性，也即根据输入参数来完成过程，而非简单地把一段可能重复使用的代码定义为函数调用。例如在本题中的move_xy函数，我自定义的该函数就是根据输入的x，y坐标来实现盘子的移动，相对而言普适性较高，在多个菜单项中也均可以使用。

5.4.2. 函数的完善

根据后续的不同要求，在函数的补充完善过程中，尽量不要使用直接复制添加一段原来的代码这种方法，会导致后续使用和维护的不便。更合理的方法是通过添加参数来实现对于不同要求对应执行部分的选择，如本题中的vertical_output函数，在之前的题目中用于输出竖排数字，之前的题目x,y坐标的值是直接在对应位置赋值的，但后续由于又要画动画演示，该打印位置需要进行改变，此次我使用的是复制原代码，修改对应位置的x,y的值，并通过定义一个新的选择变量m进行选择。但此处修改更好的选择是，添加参数x,y，以及柱子间的间隔量，根据x,y和间隔量来实现对竖排数字的打印。这样函数更具有普适性。可以实现通过修改参数来修改函数的输出效果而不必为新要求再添加代码了。

6. 附件：源程序

```
void hanoi(int n, char src, char tmp, char dst, int choice_menu)
{
    if (n == 1) {
        menu_choice(n, src, tmp, dst, choice_menu);
    }
    else {
        hanoi(n - 1, src, dst, tmp, choice_menu);
        menu_choice(n, src, tmp, dst, choice_menu);
        hanoi(n - 1, tmp, src, dst, choice_menu);
    }
}
void menu_choice(int n, char src, char tmp, char dst, int choice_menu)
{
    if (choice_menu == 1) {
        cout << setw(2) << n << "# " << src << " --> " << dst << endl;
    }
    else if (choice_menu == 2) {
        cout << setw(5) << i++ << ":" << setw(3) << n << "# " << src << "-->" << dst << endl;
    }
    else if (choice_menu == 3) {
        total_hanoi(n, src, dst, 0);
    }
    else if (choice_menu == 4) {
        total_hanoi(n, src, dst, 1);
        time_delay();
    }
    else if (choice_menu == 7) {
        total_hanoi(n, src, dst, 3);
        time_delay();
    }
    else if (choice_menu == 8) {
        total_hanoi(n, src, dst, 2);
        time_delay();
    }
}
void draw_pillar(int x, int y, int distance, int l, int h)
{
    int X = x;
    //底板输出
    for (int i = 0; i < l; i++) {
        cct_showch(x, y, ' ', COLOR_HYELLOW, COLOR_WHITE);
        x++;
        Sleep(50);
    }
    int mid1 = (X + x) / 2;
    x = x + distance;
    X = x;
    int mid2 = (X + x) / 2;
    x = x + distance;
    X = x;
    int mid3 = (X + x) / 2;
    //竖柱输出
    int Y = y;
```

```
    for (int i = 0; i < h; i++) {
        cct_showch(mid1, Y, ' ', COLOR_HYELLOW, COLOR_WHITE);
        Y--;
        Sleep(50);
    }
    Y = y;
    Y = y;
    cct_setcolor(); //恢复缺省颜色
}
void draw_plate(int x, int y, int l, int color, int b)
{
    for (int i = 0; i < l; i++) {
        cct_showch(x, y, ' ', color, COLOR_WHITE);
        x++;
        if (b == 1)
            Sleep(25);
    }
}
void vertical_output(int output1[], int output2[], int output3[],int m)
{
    if (m == 0) {
        int x, y;
        //清除原来的打印
        x = 19;
        y = 11;
        for (int t = 9; t >= 0; t--) {
            cct_gotoxy(y, x);
            cout << " ";
            x--;
        }
        x = 19;
        y = 21;
        x = 19;
        y = 31;
        //先打印A柱
        x = 19;
        y = 11;
        for (int t = 9; t >= 0; t--) {
            if (output1[t] != 0) {
                cct_gotoxy(y, x);
                cout << output1[t] << " ";
                x--;
            }
        }
        //再打印B柱
        x = 19;
        y = 21;
        //打印C柱
        x = 19;
        y = 31;
    else if (m == 1) {
    }
}
void horizontal_out(int n, char src, char dst, int die, int f)
{
    if (die == 1)
        cct_gotoxy(20, 25);
```

```
    cout << "第" << setw(4) << i++ << " 步(" << setw(2) << n << ": " << src << "-->" << dst << ") ";
}
void update_abc(int n, char src, char dst)
{
    if (src == 'A' || src == 'a') {
        a[n - 1] = 0;
        if (dst == 'B' || dst == 'b')
            b[n - 1] = n;
        if (dst == 'C' || dst == 'c')
            c[n - 1] = n;
    }
    else if (src == 'B' || src == 'b') {
    }
    else if (src == 'C' || src == 'c') {
    }
}
void total_hanoi(int n, char src, char dst,int choice)
{
    int x1 = 0, y1 = 0;
    if (choice == 1) {
        update_abc(n, src, dst);
        if (i == 1) {
            cct_gotoxy(20, 25);
            cout << "初始:                ";
            output_line(a, b, c);
            if (speed == 0)
                cin.ignore(63353, '\n');
        }
        horizontal_out(n, src, dst, 1, 1);
        if (choose_abc == 1)
            output_line(a, b, c);
        if (speed == 0)
            cin.ignore(63353, '\n');
        vertical_output(a, b, c, 0);
    }
    else if (choice == 0) {
        update_abc(n, src, dst);
        horizontal_out(n, src, dst, 0, 0);
        output_line(a, b, c);
    }
    else if (choice == 2) {
        update_abc(n, src, dst);
        cct_setcolor();
        cct_getxy(x1, y1);
        cct_gotoxy(10, 40);
        if (i == 0) {
            cout << "初始:                ";
            output_line(a, b, c);
        }
        else {
            horizontal_out(n, src, dst, 0, 0);
            output_line(a, b, c);
        }
        vertical_output(a, b, c, 1);
        cct_gotoxy(x1, y1);
        move(n, src, dst);
    }
```

```
    else if (choice == 3) {
        update_abc(n, src, dst);
        if (i == 1) {
            cct_setcolor();
            cct_getxy(x1, y1);
            cct_gotoxy(3, 25);
            horizontal_out(n, src, dst, 0, 0);
            cct_gotoxy(x1, y1);
            move(n, src, dst);
        }
    }
}
void output_number(int print_out[])
{
    int m = 0;//处理两位数与个位数打印的区别
    for (int t = 9; t >= 0; t--) {
        if (print_out[t] != 0) {
            if (print_out[t] != 10)
                cout << print_out[t] << " ";
            if (print_out[t] == 10) {
                cout << print_out[t] << " ";
                m = 1;
            }
        }
    }
    for (int t = 0; t < 10; t++) {
        if (print_out[t] == 0)
            cout << "  ";
    }
    if (m == 0)
        cout << " ";
}
void output_line(int a[10], int b[10], int c[10])
{
    cout << " A:";
    output_number(a);
    cout << "B:";
    output_number(b);
    cout << "C:";
    output_number(c);
    cout << endl;
}
void move(int n,char src ,char dst)
{
    //先确定移动的坐标数据
    int X_begin = 0, X_end = 0, Y_begin = 0, Y_end = 0;
    //初始状态部分
    Y_begin = y;
    if ((src == 'A') || (src == 'a')) {
        X_begin = mid_p1;
        for (int i = 9; i >= 0; i--) {
            if (a[i] != 0) {
                Y_begin--;
            }
            else if (a[i] = 0) {
                continue;
            }
```

```
        }
    }
    if ((src == 'B') || (src == 'b')) {
        X_begin = mid_p2;
    }
    if ((src == 'C') || (src == 'c')) {
        X_begin = mid_p3;
    }
    //结束状态部分
    Y_end = y;
    //调用坐标移动函数实现向上、向左/右、向下的过程
    char dir = 0;
    if ((src == 'A') || (src == 'a')) {
        dir = 'd';
    }
    if ((src == 'B') || (src == 'b')) {
        if ((dst == 'A') || (dst == 'a')) {
            dir = 'a';
        }
        if ((dst == 'C') || (dst == 'c')) {
            dir = 'd';
        }
    }
    if ((src == 'C') || (src == 'c')) {
        dir = 'a';
    }
    move_xy('w', X_begin, X_end, Y_begin - 1, Y_MIN, h, n);
    move_xy(dir, X_begin, X_end, Y_MIN + 1, Y_MIN + 1, h, n);
    move_xy('s', X_end, X_end, Y_MIN + 1, Y_end + 1, h, n);
}
void move_xy(char direction, int X_begin, int X_end, int Y_begin, int Y_end, int h, int n)
{
    //处理移动效果
    if (direction == 'w') {
        int Y = y - h;
        for (Y_begin; Y_begin > Y_end; Y_begin--) {
            draw_plate(X_begin - (length_plate[n-1] - 1)/2,Y_begin,length_plate[n-1],color_plate[n-1], 0);
            Sleep(20);
            if (Y_begin > Y_end + 1) {
                cct_showch(X_begin - (length_plate[n-1] - 1) / 2, Y_begin ,' ', COLOR_BLACK, COLOR_WHITE,
(length_plate[n-1] - 1) / 2);
                if (Y_begin > Y)
                    cct_showch(X_begin, Y_begin, ' ', COLOR_HYELLOW, COLOR_WHITE);
                else
                    cct_showch(X_begin, Y_begin, ' ', COLOR_BLACK, COLOR_WHITE);
                cct_showch(X_begin + 1, Y_begin, ' ', COLOR_BLACK, COLOR_WHITE, (length_plate[n-1] - 1)/2);
            }
        }
    }
    if (direction == 's') {
        int Y = y - h;
    if (direction == 'a') {
        for (X_begin; X_begin > X_end; X_begin--) {
    }
    if (direction == 'd') {
```