综合实验报告

——汉诺塔综合演示

姓　　名：　　杨恺铭

学　　号：　　2152988

专　　业：　　计算机科学与技术

1. 题目（黑体，四号，段前段后间距各1行）

题目：汉诺塔综合演示

1.1.基本解

选择汉诺塔层数n（1-10）、起始柱src和目标柱dst，并给出汉诺塔的基本解，每一步以“n# src---->dst”的形式输出

1.2.基本解（步数记录）

选择汉诺塔层数n（1-10）、起始柱src和目标柱dst，给出汉诺塔的基本解并记录步数d，每一步以“第　d 步(n# src---->dst)”的形式输出

1.3.内部数组显示(横向)

选择汉诺塔层数n（1-10）、起始柱src和目标柱dst，给出汉诺塔的基本解并、记录步数d并以横向输出的形式给出三个柱子从下到上圆盘的编号，每一步以“第　d 步(n# src---->dst) A:…B:…C:…”的形式输出

1.4.内部数组显示(纵向+横向)

选择汉诺塔层数 n（1-10）、起始柱 src、目标柱 dst 和延时 0-5（0-按回车单步演示，1-延时最长，5-延时最短），纵向给出三个柱子从下

到上圆柱的编号，在其下第四行以横向给出汉诺塔的基本解并、记录步数d并以横向输出的形式给出三个柱子从下到上圆盘的编号，每一步以“第　d 步(n# src---->dst) A:…B:…C:…”的形式输出

1.5.图形解-预备-画三个圆柱

用亮黄色画出三个圆柱分别代表A、B、C三个柱子，横向长度23，纵向长度12

1.6.图形解-预备-在起始柱上画n个盘子

选择汉诺塔层数n（1-10)、起始柱src、目标柱dst，在1.5中的三个圆柱中选择起始柱src在其上画出n个由下到上从大到小不同颜色的圆盘

1.7.图形解-预备-第一次移动

在1.6的基础上实现第一步的移动操作（将起始柱src最上面的盘子移到第一步对应的目标柱的最下面）

1.8.图形解-自动移动版本

在实现1.4功能的同时增加动画演示，制作每一步圆盘的移动动画

1.9.图形解-游戏版

每一步移动输入起始柱和目标柱，并将圆盘从输入的起始柱移动

到目标柱上，横向纵向输出需与动画演示的移动结果相匹配

2. 整体设计思路

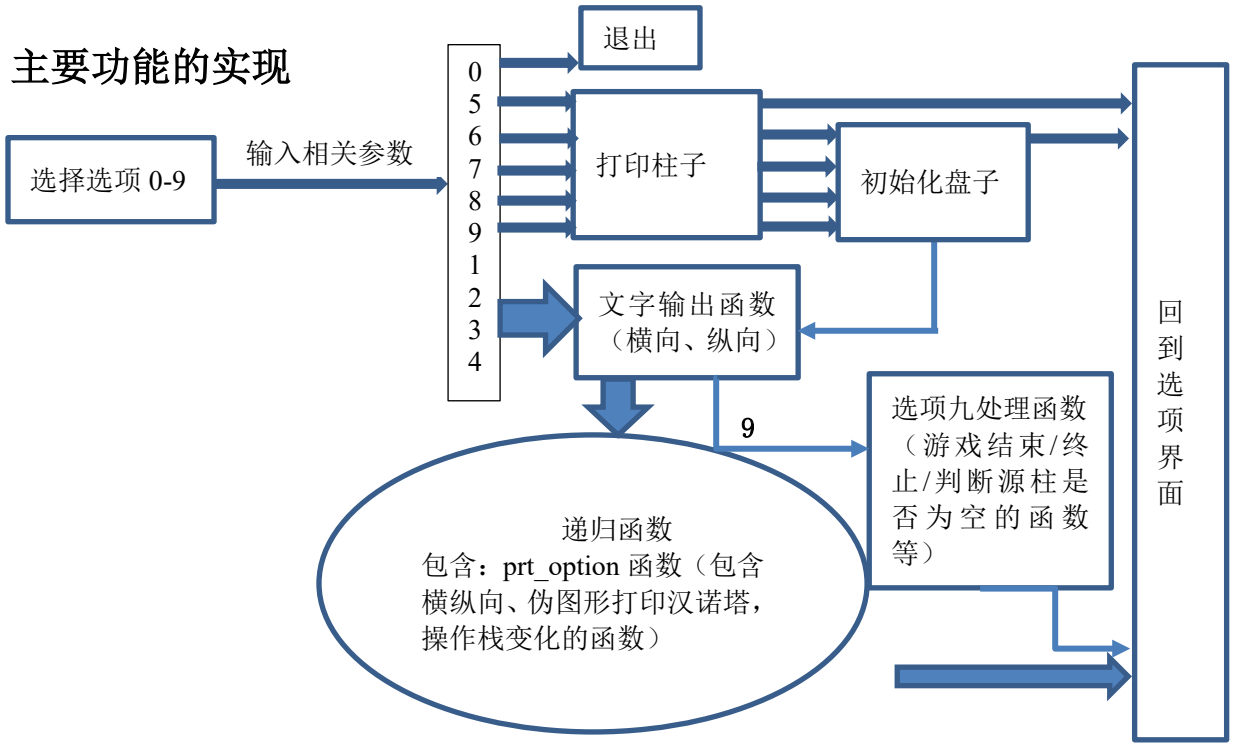
本次汉诺塔的核心部分是主函数的调用情况，柱子的打印，盘的移动过程。

柱子的打印是静态的，但有些变量可以设置成参数、变量形式。这么做的好处一个是直观，知道每一步是再干什么，还有一个是对后续的打印有帮助（比如柱子a，b，c位置），或者通过参数调整位置。参数变量包括，打印位置，柱子高度，柱子间间隔，由此可以确定：柱子a，b，c位置。

关于盘的移动过程，和纵向打印汉诺塔的本质是一回事。其逻辑上的表现就是记录汉诺塔的栈内部情况发生变化。而颜色，大小，位置都可以根据栈的情况唯一确定，所以移动的核心就是将栈的情况和伪图形的移动关联起来。

对于栈的选择，采用了二维数组+一维数组的方式。这样做可以保证传入的src，dst可以直接转换为对应柱子的012，方便书写。

3. 主要功能的实现



4. 调试过程碰到的问题

（1）进行盘子左右移动时，发现移动异常。Eg：先在（10,11）处用黑色抹去，在（11,11）处打印宽度为3的盘子。然而结果是在（13,11）处用黑色抹去，在（16,11）处打印宽度为3的盘子。按理说应该是showch的x坐标有误，但检查发现并没有什么问题。

```
//右移（左移能否用相同的地方）
for (int i = 0; i < DST_X - SRC_X; i++) {
    cct_showch(SRC_X - n + i, (SRC_Y - (HEIGHT - top[src - 'A']))
        , ' ', COLOR_BLACK, COLOR_WHITE, 2 * n + 1);

    cct_showch(SRC_X - n + i + 1, (SRC_Y - (HEIGHT - top[src - 'A']))
        , ' ', n, n, 2 * n + 1);
    if (prt_speed)
        Sleep(SLEEP / prt_speed/3 / 20);
    else
        Sleep(SLEEP/3 / 20);
}
```

后来发现是y坐标的计算错误，导致y<0,导致了showch出现异常。
所以x处的情况与预期不符不一定就是x的问题。确认x无误应该查看其他是否有问题

（2）运行每一个选项时会出现第一次运行正确，第二次运行数组和步数混乱的情况。解决方法：观察步数混乱情况，可以看到若第一次结束为第七步则第二次会从第八步开始。得到结论全局计数变量和栈没有恢复初值。
解决方法就是在每个option执行完后增加一个函数，恢复全局变量为初值。
由此可见全局变量带来的问题。若使用全局变量，且反复使用全局变量时，务必在main函数的书写中就考虑其恢复初值的状态。

5. 心得体会

5.1.设计较为复杂的程序时，分为若干小题的形式好。

首先能让我们对一个复杂程序进行剖析和分解，不至于无从下手。

其次，将其进行模块化封装，这个思想和数字逻辑课上自顶向下的设计思路有相同的内涵。其一是了解程序的整体框架，如果程序出问题了，知道应该去哪里找bug。其二是一个模块测试稳定后，就尽量不要对他进行修改，造成新的混乱。使用若干函数对其进行封装是一个很好的方法。而模块化封装的表现就是main函数，hanoi函数里面全部都是一个一个函数的调用，而没有其他多于的语句。

5.2. 关于代码的复用

代码复用的关键我觉得还是对复杂程序进行剖析和分解。将其分解成模块的过程就是将相似的功能归类，这样才能复用。

然而，分解和剖析的过程并不是那么容易。不仅要将整个程序的要求都看一遍，看到各个小题之间的联系，而且一次的构建也不可能完美，很多时候都是写着写着才发现一些更细节的东西。（很多时候还是照着老师要求做，并没有自己构建的过程）

5.3. 不足

对于hanoi函数，为了能控制纵向打印汉诺塔的位置，我只能增加汉诺塔的参数。（纵向打印汉诺塔的函数是在hanoi里面调用）然而，这些参数在整个汉诺塔过程中都是不变的，但参数的单项传值会浪费空间。不知道有没有其他更好的方法？

由于之前的小作业就用了全局const变量表示打印位置。但这样的参数一多就容易混乱。比如纵向打印汉诺塔的位置参数是PRTX，PRTY，伪图形位置参数就是SETX，SETY。刚开始写的时候就有些混乱。只能想到的解决方法是写注释。说明对变量的命名还是需要一套规则来实现比较好。比如常量都用大写，之类的。

6. 附件：源程序

1 main 函数关键部分：

```
        else if (option == 8) {
            int n;
            char src, dst, tmp;
            input(&n, &src, &dst, &tmp, 1);
            int PRT_X = 0, PRT_Y = 25;
            stack_init(src, n);
            cct_cls();
            prt_init_stack(PRT_X, PRT_Y);
            prt_stack_verticle(PRT_X, PRT_Y);
            prt_column();
            prt_init_column(src, n);
            pause();
            hanoi(n, src, tmp, dst, PRT_X, PRT_Y, option);
            to_be_continued(NULL);
            stack_clear();

        }
        else if (option == 9) {
            int n;
            char src, dst, tmp;
            input(&n, &src, &dst, &tmp, 0);
            int PRT_X = 0, PRT_Y = 25;
            stack_init(src, n);
            cct_cls();
            prt_init_stack(PRT_X, PRT_Y);
```

```
            prt_stack_verticle(PRT_X, PRT_Y);
            prt_column();
            prt_init_column(src, n);
            Sleep(500);
            game(PRT_X, PRT_Y, src, dst, n);
            to_be_continued(NULL);
            stack_clear();

        }
        else if (option == 0) {
            break;
        }
        else {
            cct_cls();
            cout << "option error" << endl;
        }
        stack_clear();
    }
```

2 菜单函数

```
int menu(void)
{
    cout << "--------------------------------" << endl;
    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:] ";
    int i;
    while (1) {
        i = _getche();
        if (i >= '0' && i <= '9') {
            break;
        }
        else {
            cout << "\b \b";
        }
    }
    return i;
}
```

3 主要函数：

汉诺塔递归函数

```
void hanoi(int n, char src, char tmp, char dst, int PRT_X, int PRT_Y, int option)
{
    if (n == 0)
        return;
    hanoi(n - 1, src, dst, tmp, PRT_X, PRT_Y, option);
    cnt++;
    operate(src, dst);
```

```
    prt_option(n, src, tmp, dst, PRT_X, PRT_Y, option);
    hanoi(n - 1, tmp, src, dst, PRT_X, PRT_Y, option);
}
void prt_option(int n, char src, char tmp, char dst, int PRT_X, int PRT_Y, int option)
{
    if (option == 1) {
        cout << n << "# " << src << "---->" << dst << endl;
    }
    if (option == 2) {
        cout << "第" << setw(4) << cnt
            << "步(" << n << "#: "
            << src << "-->" << dst << ")"<<endl;

    }
    if (option == 3) {
        cout << "第" << setw(4) << cnt
            << "步(" << n << "#: "
            << src << "-->" << dst << ")";
        prt_stack();
        cout << endl;
    }
    if (option == 4 || option == 8) {
        prt_stack_horizontal(PRT_X, PRT_Y, src, dst, n);
        prt_stack_verticle(PRT_X, PRT_Y);
        operate(dst, src);
        if (option == 8)
            prt_move(src, dst, n);
        operate(src, dst);
        pause();
    }
}
```

伪图形打印移动过程

```
void prt_move(const char src, const char dst, const int n, int SETX, int SETY, int
flag_game)
{
    const int SRC_X = SET_COL_A + (src - 'A') * (LENGTH + GAP),
        SRC_Y = SETY - top[src - 'A'];
    const int DST_X = SET_COL_A + (dst - 'A') * (LENGTH + GAP),
        DST_Y = SETY - top[dst - 'A'];
    //上升
    for (int i = 0; i < HEIGHT - top[src - 'A']; i++) {
        cct_showch(SRC_X - n, SRC_Y - i, ' ', COLOR_BLACK, COLOR_WHITE, 2 * n + 1);
        cct_showch(SRC_X, SRC_Y - i, ' ', COLOR_WHITE, COLOR_WHITE, 1);
        cct_showch(SRC_X - n, SRC_Y - i - 1, ' ', n, n, 2 * n + 1);
        if (prt_speed)
            Sleep(SLEEP/prt_speed / 20);
        else
            Sleep(SLEEP / 20);
    }

    if (src < dst) {
        //右移（左移能否用相同的地方）
        for (int i = 0; i < DST_X - SRC_X; i++) {
            cct_showch(SRC_X - n + i, (SRC_Y - (HEIGHT - top[src - 'A']))
                , ' ', COLOR_BLACK, COLOR_WHITE, 2 * n + 1);
```

```
            cct_showch(SRC_X - n + i + 1, (SRC_Y - (HEIGHT - top[src - 'A']))
                , ' ', n, n, 2 * n + 1);
            if (prt_speed)
                Sleep(SLEEP / prt_speed/3 / 20);
            else
                Sleep(SLEEP/3 / 20);
        }
    }
    else {
        //左移
        for (int i = 0; i < SRC_X - DST_X; i++) {
            cct_showch(SRC_X - n - i, (SRC_Y - (HEIGHT - top[src - 'A']))
                , ' ', COLOR_BLACK, COLOR_WHITE, 2 * n + 1);

            cct_showch(SRC_X - n - i - 1, (SRC_Y - (HEIGHT - top[src - 'A']))
                , ' ', n, n, 2 * n + 1);
            if (prt_speed)
                Sleep(SLEEP / prt_speed/3 / 20);
            else
                Sleep(SLEEP/3 / 20);
        }
    }


    //下降
    for (int i = HEIGHT - top[dst - 'A']; i > 1; i--) {
        cct_showch(DST_X - n, DST_Y - i, ' ', COLOR_BLACK, COLOR_WHITE, 2 * n + 1);
        if (i != HEIGHT - top[dst - 'A']) {
            cct_showch(DST_X, DST_Y - i, ' ', COLOR_WHITE, COLOR_WHITE, 1);
        }
        cct_showch(DST_X - n, DST_Y - i + 1, ' ', n, n, 2 * n + 1);
        if (prt_speed)
            Sleep(SLEEP / prt_speed/20);
        else
            Sleep(SLEEP/20);
    }
    cct_setcolor();
}
```

横纵向打印汉诺塔

```
void prt_stack_horizontal(int PRT_X, int PRT_Y, char src, char dst, int n)
{
    cct_gotoxy(PRT_X, PRT_Y + 4);
    cout << "第" << setw(4) << cnt
        << "步(" << n << "#: "
        << src << "-->" << dst << ")";
    prt_stack();
}

void prt_stack_verticle(int PRT_X, int PRT_Y)
{
    //恢复默认颜色
    int xa = PRT_X + 2, xb = xa + (PRT_LEN - 5) / 2,
        xc = xa + (PRT_LEN - 5);
    cct_gotoxy(PRT_X, PRT_Y);
    for (int i = 0; i < PRT_LEN; i++) {
        cout << '=';
```

```
    }
    cct_gotoxy(xa, PRT_Y + 1);
    cout << "A" << setw((PRT_LEN - 5) / 2)
        << "B" << setw((PRT_LEN - 5) / 2) << "C";

    for (int set = 0; set < N_STACK; set++) {
        for (int i = 0; i < N; i++) {
            cct_gotoxy(xa + set * (PRT_LEN - 5) / 2 - 1, PRT_Y - 1 - i);
            if (i < top[set]) {
                cout << setw(2) << stack[set][i];
            }
            else {
                cout << setw(2) << "";
            }
        }
    }
}
```

游戏函数

```
void game(int PRT_X, int PRT_Y, char src, char dst, int n)
{
    cct_gotoxy(0, 0);
    cout << "从 "<<src<<" 移动到 "<<dst<<", 共 "<<n<<" 层";
    cct_gotoxy(0, PRT_Y + 6);
    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) ： ";
    int cnt = 0;
    while (1) {
        if (top[dst - 'A'] == n) {
            cct_gotoxy(0, PRT_Y + 7);
            cout << "游戏结束！！！";
            break;
        }
        char src, dst;
        while (1) {
            cct_gotoxy(61, PRT_Y + 6);
            char tmp[3];
            cin.getline(tmp, 3, '\n');
            src = tmp[0];
            dst = tmp[1];

            if (src >= 'a' && src <= 'c')
                src -= 32;
            if (dst >= 'a' && dst <= 'c')
                dst -= 32;
            if (src == 'Q'||src == 'q') {
                return;
            }
            if (cin.fail()) {
                cin.clear();
                cin.ignore(65536, '\n');
            }

            else if ((src >= 'A' && src <= 'C')
                && (dst >= 'A' && dst <= 'C') && (src != dst)) {
                if (0 == top[src - 'A']) {
                    cout << endl << "源柱为空!";
                    Sleep(500);
```

```
                }
                else if (0 != top[dst - 'A']
                    && stack[src - 'A'][top[src - 'A']-1]
                    > stack[dst - 'A'][top[dst - 'A'] - 1]) {
                    cout << endl << "大盘压小盘，非法移动!";
                    Sleep(500);

                }
                else {
                    cct_showch(61, PRT_Y + 6, ' ', 0, 7, 100);
                    cnt++;
                    break;
                }
            }
            Sleep(SLEEP/5);
            cct_showch(61, PRT_Y + 6,' ',0,7,100);
            cct_showch(0, PRT_Y + 8, ' ', 0, 7, 50);
        }
        operate(src, dst);
        prt_stack_horizontal(PRT_X, PRT_Y, src, dst, cnt);
        prt_stack_verticle(PRT_X, PRT_Y);
        operate(dst, src);
        prt_move(src, dst, stack[src - 'A'][top[src - 'A'] - 1]);
        operate(src, dst);

    }
}
```