同济大学

《高级语言程序设计》

实验报告

报告名称：汉诺塔综合演示实验报告

班　　级：信05

学　　号：2353814

姓　　名：马小龙

日　　期：2024年5月15日

1. 题目

1.1.题目描述

旧程序4-b12（汉诺塔基本移动），4-b13（步数移动），5-b6（横向每一步详细）、5-b7（横向纵向每一步详细）。

新程序：通过动画演示汉诺塔每一步详细过程，同时包含横向纵向每一步详细信息。为减小难度，程序分为5步完成，画三根柱子、画柱子和n个圆盘，实现第一步移动动画，实现演示完整过程、游戏。

将之前所有汉诺塔的相关程序即新程序集合在一个程序当中，采用菜单的方式进行需要实现的功能选择，并使用伪图形界面。

程序需要根据实现以下10个功能：

1.基本解
2.基本解(步数记录)
3.内部数组显示(横向)
4.内部数组显示(纵向+横向)
5.图形解-预备-画三个圆柱
6.图形解-预备-在起始柱上画n个盘子
7.图形解-预备-第一次移动
8.图形解-自动移动版本
9.图形解-游戏版
0.退出

1.2.题目要求

1.2.1.变量要求

除以下内容外，其他均不允许使用全局变量记录（全局const变量/#define宏定义的数量不受限制）

1.总移动步数 ：1个全局简单变量
2.圆柱上现有圆盘的编号：3个全局一维数组或1个全局二维数组
3.圆柱上现有圆盘的数量：3个全局简单变量或1个全局一维数组
4.延时 ：1个全局简单变量

1.2.2.函数要求

1.整个程序只能使用一个递归函数，递归函数不能超过15行
2.菜单1/2/3/4/6/7/8中的输入多个参数必须共用一个函数（仅该函数可以使用指针相关内容）
3.横向输出、纵向输出、画柱子、盘子移动等功能的实现需要共用一个函数
4.其他个功能实现的细则要求

1.2.3.功能实现要求

功能5:添加延时以方便观察

功能6:1）假设三根柱子从左到右依次为A、B、C，从大到小依次画圆盘，圆盘颜色各不相同，

2）添加延时以方便观察

功能7:移动需要先上移，再平移，最后下移；

功能8:移动需要先上移，再平移，最后下移；

功能9:1）每次输入为A-C(大小写均可)之间，为本次移动

2）每次输入要检查输入合理性，不合理则提示出错并重输

3）移动需要先上移，再平移，最后下移

4）所有盘子移动到目标柱时提示“游戏结束”

2. 整体设计思路

汉诺塔要直接得出n层如何从起始柱移动到目标柱较为困难。我们可以借助递归的思想，将汉诺塔问题分为以下几步，就可以简化这一问题：

1.将前n-1个盘子从起始柱移到中间柱

2.将第n个盘子从起始柱移到目标柱

3.将前n-1个盘子从中间柱移到目标柱

这样，我们只需要得到前n-1个盘子的移动便可以得到结果，而前n=n-1个盘子的移动也可以依据这种方法简化，直到n=1，而对于一个盘子的移动，显然十分简单。

以上方法可以让我们了解汉诺塔每一步该如何移动，但无法确定每一步时每一个柱子上有几个盘子，分别是几号盘子，而面对问题，我们可以将汉诺塔每一个柱子看成一个栈，盘子进来或出去看作进栈或出栈。进而我们可以将栈抽象化，将一个栈看作一个一维数组，其中用数组中元素的值来表示盘子的编号，即1-9，元素的值为0来表示此时没有盘子。

不过，此时依旧无法判断柱子上有多少盘子，为此，我们引进一个新的变量，来表明柱子上有多少盘子，从而就可以确定每一个柱子的状态，进而就可以完整的表示柱子的状态，能够确定每一步横向输出、纵向输出，及画图的内

3. 主要功能的实现

3.1.重要功能的实现过程

3.1.1.汉诺塔基本移动

设有n个盘子，执行hanoi(n)，先将前n-1个盘子移动到中间柱，即执行hanoi(n-1)，再将第n个盘子移动到目标柱，最后将前n-1个盘子从中间柱移动到目标柱，即执行(n-1)。由此类推，即可以实现汉诺塔的基本移动过程。程序执行时，需注意实际移动的起始柱和目标柱。

3.1.2.每一次移动柱子状态确定

开始时定义起始柱和层数n时，我们可以对起始柱对应数组及层数确定变量赋值，对数组依次赋n，n-1，……直到1，层数则为盘的个数n，中间柱、目标柱对应数组、变量赋值为0。在后续每一步移动中，确定当下移动的起始柱和目标柱后，他们对应状态变化为将起始柱对应数组最后一个非零元素赋值给目标柱对应数组第一个零元素，同时给起始柱对应数组最后一个非零元素重新赋值变为零，起始柱层数-1，目标柱层数+1.这样，就是实现了每一次移动柱子状态的确定。需注意，起始柱和目标柱时对当下这一步而言。

确定了每一次移动柱子的状态，横向和纵向输出便很容易解决。

3.1.3.柱子和盘子的绘制

实现图形绘制，可以调用cct_showint(X, Y, num, bg_color, fg_color, rpt)函数，实际上就从（X,Y）开始，用指定颜色输出长度为rep的字符串，多次调用可以实现在该处画出某一图形的效果，不过，由于设置背景色会对后续输出产生影响，必要时需要设置回初始状态。

3.1.4.盘子移动

盘子移动可以通过调用cct_showint(X, Y, num, bg_color, fg_color, rpt)函数，一面用系统默认颜色输出长度为rep的字符串来清除盘子，一面用指定颜色输出长度为rep的字符串来打印盘子，需注意，若是清除了柱子，需要重新打印出来。这样，就在视觉上形成了盘子移动的效果。

3.2.主要函数及其功能

- void input(char* src, char* dst, char* tmp, int* n, char order)

  负责输入起始柱、目标柱以及汉诺塔层数，以及功能四和功能八延时所需变量，并判断得出中间柱

- void hanoi(int n, char src, char tmp, char dst,char order)

  通过递归函数，负责汉诺塔每一步移动过程的输出

- void step(int src, int dst)

  汉诺塔每一步移动后，执行该函数，通过数组值得变化和非零元素个数的变化来模拟每个柱子上盘子的变化

- void landscape() & portrait(int order)

  分别负责横向输出和纵向输出

- void column() & void disc(char src,int n)

通过调用cct_showint(X, Y, num, bg_color, fg_color, rpt)函数，分别负责打印柱子和盘子

- void display(char src,char dst)

  通过调用cct_showint(X, Y, num, bg_color, fg_color, rpt)函数，实现每一步盘子移动的动画

- void game(int n, char src, char tmp, char dst, char order)

  服务于功能9，负责游戏每一步的输入与输出，同时进行错误处理以及调用其他函数。

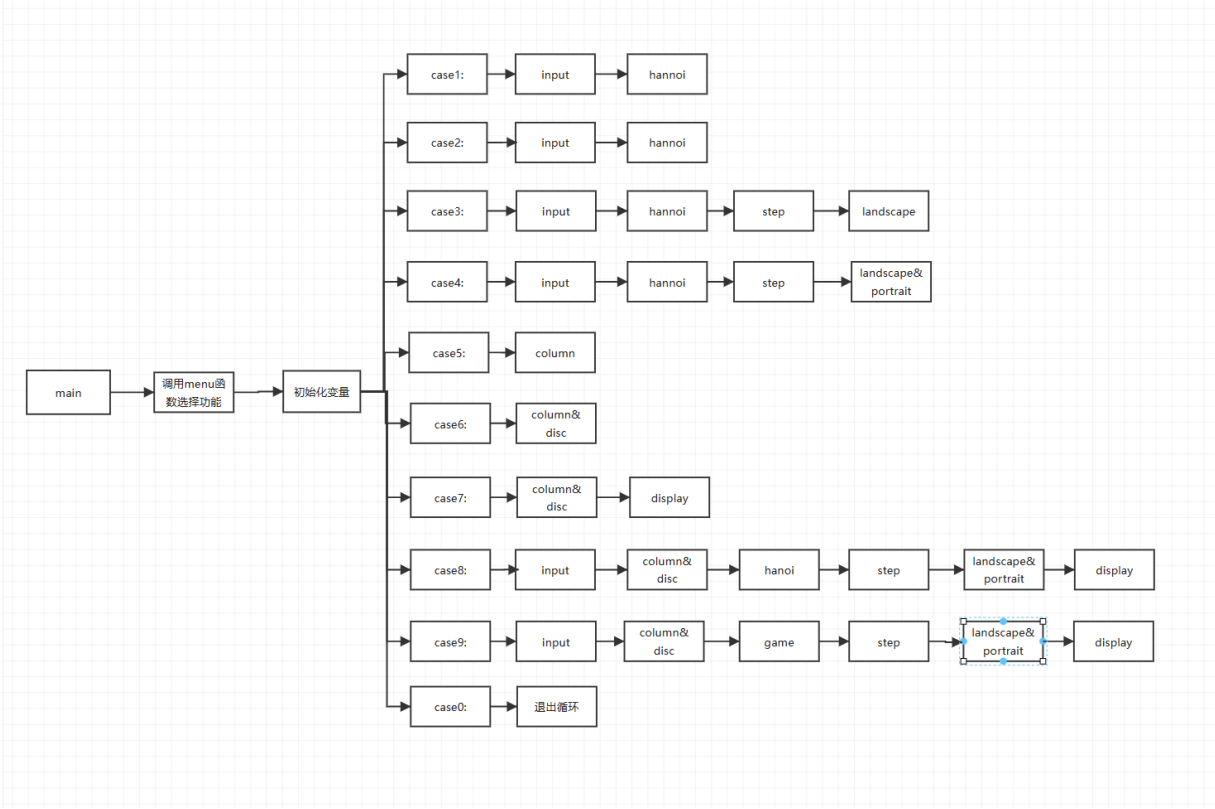
- void speed()

  根据参数，设置延时

- void initialize()

  初始化全局变量

3.3. 流程框图



4. 调试过程碰到的问题

4.1.柱子消失问题

在实现盘子的移动过程中，我通过调用cct_showint(X, Y, num, bg_color, fg_color, rpt)函数依次清除已经打印过的盘子、再次在坐标变化为1的位置打印该盘子，但总是会出现原本的柱子残缺或消失的情况。

通过思考，我认为是在清除盘子的过程中顺带将柱子也清除了，或者说，在我打印盘子的时候柱子对应位置的颜色已经发生了变化，也就是柱子消失，在清除盘子时，盘子部分的颜色恢复为系统默认颜色，因此出现柱子残缺或消失的情况。

为了解决这一问题，我在清除已经打印过的盘子后再次调用cct_showint(X, Y, num, bg_color, fg_color, rpt)函数，将消失的柱子再次打印出来，柱子便不再残缺。

4.2.游戏输入问题

在游戏过程中输入移动的起点与终点时，demo表现出以下特性：

1.输入一定长度会停止输入，清除显示的内容，开始重新输入

2.可以读入回车但不会显示，不会读入和显示空格等字符

3.已经输入的内容无法被删除

尤其是第三点，已学的输入都无法满足这一要求，后来我想到，显示出来的字符是否是输出的内容而不是因输入而显示的内容。于是，我选择定义一个大小为20的数组，并选择没有回显的_getch()来读取输入的字符，给数组输入值，同时，对于数组元素值的范围做出限制，不满足则重新输入，最后输出该元素的值，如此循环20次，这样，就可以实现与demo相似的输入功能。

4.3.游戏的清除问题

游戏中，每一次输入的内容或者错误信息提示会在下一次输入时清除，开始时，我无法做到这一点，总是在原来已经输出过的内容输出新的内容，这会使得界面会有多余内容。

为了解决这一问题，我借鉴了打印汉诺塔柱子和盘的思路了，调用cct_showint(X, Y, num, bg_color, fg_color, rpt)函数，每一步结束后在改变输出的区域的颜色，使输出内容消失，然后设定初始颜色，不影响下一次输入。

5. 心得体会

5.1.经验与收获

在本次作业完成的过程中，对于某一程序中的类似部分，我进行了复制粘贴，但忘记了根据具体情况修改函数的值，导致在调试过程中总会出现错误，而自己也很难发现究竟是哪里的错误，致使自己花费了很长时间来调试。因此，对于相似的程序，要复制粘贴的话一定要记着根据情况修改内容，或者直接写而非复制粘贴。

此外，本次程序我也出现了数组越界问题，主要是条件错误，导致了越界，以后写程序也应当注意

次问题。

最后，通过完成本次程序，我学习了很多新的函数，对光标改变、颜色改变的函数有了较为深刻的理解，并能熟练应用。

5.2.复杂程序分成若干小题好还是一道大题好

在做一道复杂的大题时，我认为分为若干小题较好。

首先，将复杂的大题分解成较小的、可管理的小题可以降低问题的复杂性，使每个部分更加易于理解和解决。其次，逐步推进的方法有助于保持思路清晰，不至于被问题的整体复杂性压倒。再者，集中精力于一个个具体问题，而不是去考虑所有的复杂因素，可以提高效率。还有，若出现错误，只需修改相关小题，而不必重新思考整个大题。最后，分解成若干小题意味着分数的分离，可以完成小题以获得对应分数，这就不会使自己在难以完成整个程序时过于焦虑，学习高程的信心不会被打倒。

5.3.总结

在编写复杂程序时，我仔细考虑了程序各部分之间的关系，尤其是横向输出、纵向输出和动画的实现。这些功能的实现原理基本相同，都是在不同的形式下输出当前柱状态的表现。实现的关键在于确认每一步的柱的状态，我通过一个核心函数step()来计算和更新柱的状态，而这一函数被多次调用，实现了重用代码，使代码不再冗杂。

为了更好地重用代码，需要深入考虑每个功能之间的内在联系。通过仔细分析，我们可以找到它们之间的共通之处，这样一来，代码就可以被尽可能高效地利用，避免重复编写相似的代码片段。

5.4.思考

要编写复杂的程序，首先需要将复杂的程序逐步分解为多个简单的程序模块。这样做可以提高程序完成的可能性，减少程序的复杂性。分解的过程不仅可以让我们更清晰地理解每个模块的功能和实现细节，还能使我们更容易地发现和修复潜在的错误。

在分解复杂程序时，我们要善于归纳总结，考虑不同小程序之间的逻辑关系，寻找它们之间的联系和共通之处。通过这种方式，我们可以高效地重用代码，避免每个小程序之间相互独立，从而减少代码的冗杂。这不仅提高了代码的可维护性，还提升了代码的可读性和可扩展性。

另外，在编写程序时，应该始终关注模块化和抽象化。模块化设计使得每个小程序具有独立的功能，方便进行测试和调试。而抽象化则帮助我们在更高的层次上思考问题，避免陷入具体实现的细节之中。通过合理的模块化和抽象化，我们可以更容易地进行代码的复用和扩展。

6. 附件：源程序

```
void speed()
{
    if (velocity == 0) {
        while (_getch() != 13)
            ;
```

```
    }
    else if (velocity == 1)
        Sleep(200);
    else if (velocity == 2)
        Sleep(150);
    else if (velocity == 3)
        Sleep(100);
    else if (velocity == 4)
        Sleep(50);
    else
        Sleep(0);
}
void column()
{
    cct_showch(1, 15, ' ', 14, 14, 23);
    cct_showch(33, 15, ' ', 14, 14, 23);
    cct_showch(65, 15, ' ', 14, 14, 23);
    for (int i = 1; i <= 12; i++) {
        cct_showch(12, 15-i, ' ', 14, 14, 1);
        Sleep(50);
        cct_showch(44, 15-i, ' ', 14, 14, 1);
        Sleep(50);
        cct_showch(76, 15-i, ' ', 14, 14, 1);
        Sleep(50);
    }
    cct_setcolor(0, 7);
}
void disc(char src,int n)
{
    if (src == 65) {
        for (int i = n; i > 0; i--) {
            cct_showch(12 - i, 14 + i - n, ' ', i, i, 2 * i + 1);
            Sleep(50);
        }
    }
    if (src == 66) {
        for (int i = n; i > 0; i--) {
            cct_showch(44 - i, 14 + i - n, ' ', i, i, 2 * i + 1);
            Sleep(50);
        }
    }
    if (src == 67) {
        for (int i = n; i > 0; i--) {
            cct_showch(76 - i, 14 + i - n, ' ', i, i, 2 * i + 1);
            Sleep(50);
        }
    }
    cct_setcolor(0, 7);
}
void display(char src,char dst)
{
    int X_start, Y_start, X_end, Y_end, pan;
    if (dst == 65) {
        X_end = 12;
        Y_end = topA;
        pan=A[topA-1];
```

```
    }
    /* B、C与A相似,省略*/
    if (src == 65) {
        X_start = 12;
        Y_start = topA+1;
    }
   /* B、C与A相似,省略*/
    cct_gotoxy(0, 38);
    for (int i = 15-Y_start; i > 1; i--) {
        cct_showch(X_start-pan, i, ' ', 0, 7, 2 * pan + 1);
        if(i>=3)
            cct_showch(X_start, i, ' ', 14, 14, 1);
        if (velocity != 5)
            Sleep(50);
        cct_showch(X_start-pan, i-1, ' ', pan, pan, 2 * pan + 1);
        if (velocity == 0)
            Sleep(50);
        else
            speed();
    }
    if (X_start < X_end) {
        for (int j = X_start; j < X_end; j++) {
            cct_showch(j-pan, 1, ' ', 0, 7, 2 * pan + 1);
            if (velocity != 5)
                Sleep(50);
            cct_showch(j-pan+1, 1, ' ', pan, pan, 2 * pan + 1);
            if (velocity == 0)
                Sleep(50);
            else
                speed();
        }
    }
    else if (X_start > X_end){
        for (int j = X_start; j > X_end; j--) {
            cct_showch(j-pan, 1, ' ', 0, 7, 2 * pan + 1);
            if (velocity != 5)
                Sleep(50);
            cct_showch(j-pan-1, 1, ' ', pan, pan, 2 * pan + 1);
            if (velocity == 0)
                Sleep(50);
            else
                speed();
        }
    }
    for (int k =1 ; k < 15-Y_end; k++) {
        cct_showch(X_end - pan, k, ' ', 0, 7, 2 * pan + 1);
        if(k>=3)
            cct_showch(X_end, k, ' ', 14, 14, 1);
        if (velocity != 5)
            Sleep(50);
        cct_showch(X_end - pan, k + 1, ' ', pan, pan, 2 * pan + 1);
        if (velocity == 0)
            Sleep(50);
        else
            speed();
    }
    cct_setcolor(0, 7);
```

```
}
void step(int src, int dst)
{
    int pan;
    if (src == 65) {
        pan = A[topA - 1];
        A[topA - 1] = 0;
        topA = topA - 1;
    }
    /* B、C与A相似,省略*/
    if (dst == 65) {
        A[topA] = pan;
        topA++;
    }
    /* B、C与A相似,省略*/
}
void landscape()
{
    int i;
    cout << " A:";
    for (i = 0; i < 10; i++) {
        if (A[i] != 0)
            cout << setw(2) << A[i];
        else
            cout << "  ";
    }
    cout << " B:";
    for (i = 0; i < 10; i++) {
        if (B[i] != 0)
            cout << setw(2) << B[i];
        else
            cout << "  ";
    }
    cout << " C:";
    for (i = 0; i < 10; i++) {
        if (C[i] != 0)
            cout << setw(2) << C[i];
        else
            cout << "  ";
    }
}
void portrait(char order)
{
    int Y_zuobiao;
    if (order == '4')
        Y_zuobiao = 12;
    else
        Y_zuobiao = 27;
    cct_gotoxy(9, Y_zuobiao);
    cout << "========================";
    cct_gotoxy(9, Y_zuobiao+1);
    cout << "  A         B         C  ";
    int i;
    for (i = 0; i < 10; i++) {
        cct_gotoxy(10, Y_zuobiao-1 - i);
        if (A[i] != 0)
            cout << setw(2) << A[i];
```

```
        else
            cout << "  ";
    }
    for (i = 0; i < 10; i++) {
        cct_gotoxy(20, Y_zuobiao-1 - i);
        if (B[i] != 0)
            cout << setw(2) << B[i];
        else
            cout << "  ";
    }
    for (i = 0; i < 10; i++) {
        cct_gotoxy(30, Y_zuobiao-1 - i);
        if (C[i] != 0)
            cout << setw(2) << C[i];
        else
            cout << "  ";
    }
}
void detail(int n, char src, char tmp, char dst, char order)
{
    if(order=='1')
        cout << n << "# " << src << "---->" << dst << endl;
    else if (order == '2') {
        cout << "第" << setw(4) << num << " 步(" << setw(2) << n << "#: " << src << "-->" << dst << ")" <<
endl;
        num++;
    }
    else if (order == '3') {
        cout << "第" << setw(4) << num << " 步(" << setw(2) << n << "#: " << src << "-->" << dst << ") ";
        num++;
        step(src, dst);
        landscape();
        cout << endl;
    }
    else if (order == '4')
    {
        speed();
        cct_gotoxy(0, 17);
        cout << "第" << setw(4) << num << " 步(" << setw(2) << n << "#: " << src << "-->" << dst << ") ";
        num++;
        step(src, dst);
        landscape();
        portrait(order);
    }
    else if (order == '8'|| order == '9') {
        if (order == '8')
            speed();
        cct_gotoxy(0, 32);
        cout << "第" << setw(4) << num << " 步(" << setw(2) << n << "#: " << src << "-->" << dst << ") ";
        num++;
        step(src, dst);
        landscape();
        portrait(order);
        display(src, dst);
    }
}
```

```
void hanoi(int n, char src, char tmp, char dst,char order)
{
    if (n == 1)
        detail(n, src, tmp, dst, order);
    else {
        hanoi(n - 1, src, dst, tmp,order);
        detail(n, src, tmp, dst, order);
        hanoi(n - 1, tmp, src, dst,order);
    }
}
void game(int n, char src, char tmp, char dst, char order)
{
    char start, end;
    char shuru[20] = {0};
    int pan;
    cct_cls();
    cout << "从 " << src << " 移动到 " << dst << ", 共 " << n << " 层" << endl;
    portrait(order);
    cct_gotoxy(0, 32);
    cout << "初始: ";
    landscape();
    column();
    disc(src, n);
    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) ：";
    int x, y;
    cct_getxy(x, y);
    while (1) {
        cct_showch(x, y, ' ', 0, 0, 20);
        cct_showch(0, y + 1, ' ', 0, 0, 25);
        cct_setcolor(0, 7);
        cct_gotoxy(x, y);
        int i=0;
        while (i < 20) {
            shuru[i] = _getch();
            if (shuru[i] == 13)
                break;
            else if (shuru[i] <= 32 || shuru[i] == 127)
                continue;
            cout << shuru[i];
            i++;
        }
        if (i > 2||i==0)
            continue;
        if (i == 1) {
            if (shuru[0] == 81 || shuru[0] == 113) {
                cct_gotoxy(0, y + 1);
                cout << "游戏中止!!!!!";
                return;
            }
            else
                continue;
        }
        start = shuru[0];
        end = shuru[1];
        if (start >= 97)
            start = start - 32;
```

```
        if (end >= 97)
            end = end - 32;
        if (start < 65 || start > 67|| end < 65 || end>67||start == end)
            continue;
        if (start == 65) {
            if (topA == 0) {
                cct_gotoxy(0, y + 1);
                cout << "源柱为空!";
                Sleep(500);
                continue;
            }
        }
        /* B、C与A相似,省略*/
        if (start == 65) {
            if (end == 66) {
                if (topB>0&&A[topA - 1] > B[topB - 1]) {
                    cct_gotoxy(0, y + 1);

                    cout << "大盘压小盘，非法移动!";
                    Sleep(500);
                    continue;
                }
            }
            if (end == 67) {
                if (topC>0&&A[topA - 1] > C[topC - 1]) {
                    cct_gotoxy(0, y + 1);
                    cout << "#";
                    cout << "大盘压小盘，非法移动!";
                    Sleep(500);
                    continue;
                }
            }
            pan = A[topA - 1];
        }
        /* B、C与A相似,省略*/
        detail(pan, start, tmp,end, order);
        if (end == 65) {
            if (topA == n) {
                cct_gotoxy(0, y + 1);
                cout << "游戏结束!!!!!";
                return;
            }
        }
        /* B、C与A相似,省略*/
        }
    }
}
```