装

订

线

高级程序语言设计上机实验报告I

——汉诺塔综合演示

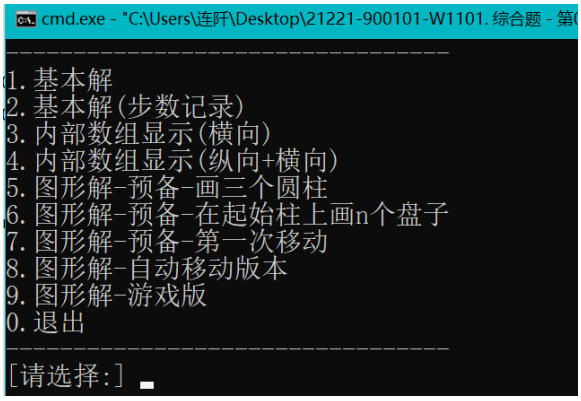
2151140 信07 王谦

2021年12月5日

1. 汉诺塔综合演示题目描述

1.1. 题目整体目标

借助提供的伪图形界面工具函数集，将多个与汉诺塔游戏有关的小题集成在一个程序中，用菜单方式选择执行。其中菜单1-4项为已经分别实现的独立小程序，5-9项为需要新实现的内容。目标效果以给出的demo为参考。



1.2. 菜单各项目标

1. 基本解答每次的移动
2. 基本解加步数记录
3. 内部数组横向显示
4. 内部数组纵向加横向显示
5. 画三个圆柱
6. 画起始柱盘子
7. 动画效果第一次移动
8. 动画效果自动解答
9. 动画效果按规则任意移动的游戏版
0. 退出

各目标效果以给出的demo为参考。

1.3. 额外的要求与限制

常规的规范限制：如不允许使用后续知识点；不允许使用goto、C++的string类等知识内容以及相关的文件命名与提交要求规范。

全局变量的限制：本次只允许定义总移动步数、现有圆盘编号、现有圆盘数量和延时四类全局变量或全局数组，且这些量也有严格限制。

递归函数的限制：本程序只允许使用一个递归函数且不允许超过15行。

函数共用限制：横向输出和纵向输出需要各共用一个函数，画柱子和移动盘子也各共用一个函数。

清屏限制：不允许每变换一次就清一次屏重输出。

1.4. 其它

字体字号、颜色、时长、提示信息具体内容、全半角、空格数等均无强制要求。

提供了一个比较全面的基本函数工具包，保证了初学者具备足够的工具实现目标效果。

2. 整体设计思路

2.1. 大致策略

除去提供的函数工具外，建立三个cpp。其中hanoi_menu.cpp中包含menu函数实现菜单界面与选择效果；hanoi_multiple_solutions.cpp中自定义需要的各类函数；hanoi_main.cpp中包含main函数调用各函数，根据调用函数的返回值进一步判断接下来需要执行那些函数或步骤；策略是将复杂的功能分解为许多更具有通用性的小函数，通过不同的组合与参数调整，实现各个菜单项的特定目标。

2.2. 举例

我自己额外设了许多函数，如图所示，借助这些小函数与提供的工具函数来实现各个菜单项的特定目标。

例如打印形象化的数字和塔上的盘子时，尤其是实现移动盘子的动画效果时，主要的实现途径就是不断的用参数调整位置输入相应的字符串，与此同时用空格覆盖掉需要清除的部分内容。

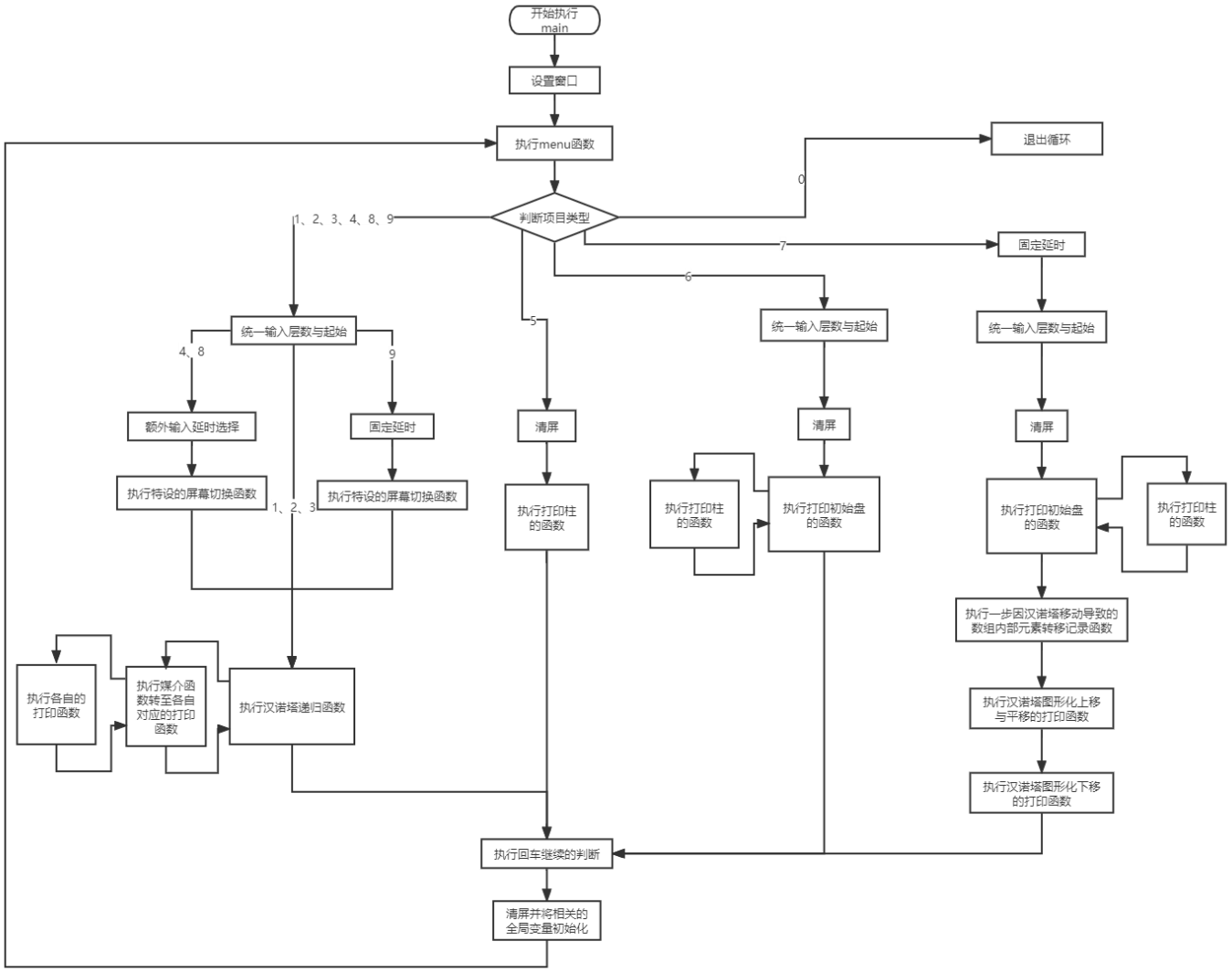
并且，在一些需要程序执行过程中手动输入的内容要注意有效地储存并使用合适的方法传递给需要的相应函数判断与执行，并在需要的时候及时清空或还原。

同时，每一个菜单项需要实现的效果都有些不同，反复调用小函数过多可能会有些乱，可以为部分菜单项专门建立一个函数把相关需要的函数直接按照顺序整合进来，只调用一个专门的菜单项函数就可以实现这一项的目标效果。这样可以显得更加清晰。

另外，注意程序的整体结构性，main函数本身即为一个大的循环结构，各个菜单项的执行则是各个分支，在每个分支中又有额外的更深层的循环分支选择结构。需要注意每一步骤对应的逻辑层次，明确循环与跳出的时机。

3. 主要功能的实现

整体流程如图所示。



其中的诸多打印函数多次使用了在指定位置特定颜色打印字符串的工具函数，清除内容时则打印空格进行覆盖；通过循环结构与参数调整就可以打印出柱子与盘子等效果。

另外，由于我的函数不够完善，在许多细节的地方都需要进行特设来达到目标效果。图中“执行各自的打印函数”就是我为不同的菜单项组合出了各自的处理函数，使用一个媒介函数放在递归函数中作为中介判断并调用相应的打印函数。这也同时极大地减少了递归函数的代码长度，正符合了题目的要求。

```
/*唯一的汉诺塔递归函数*/
void hanoi(int choose, int n, char src, char tmp, char dst)
{
    if (n == 1) {
        media(choose, n, src, tmp, dst);//cout << setw(2) << n << "# " << src << "-->" << dst << endl;
    }
    else {
        hanoi(choose, n - 1, src, dst, tmp);
        media(choose, n, src, tmp, dst);//cout << setw(2) << n << "# " << src << "-->" << dst << endl;
        hanoi(choose, n - 1, tmp, src, dst);
    }
}

/*输出中转的媒介函数*/
void media(int choose, int n, char src, char tmp, char dst)
{
    switch (choose) {
        case 1:
            printstep1(n, src, dst);
            break;
        case 2:
            printstep2(n, src, dst);
            break;
        case 3:
            printstep3(choose, n, src, dst);
            break;
        case 4:
            printstep4(choose, n, src, dst);
            break;
        case 8:
            printstep8(choose, n, src, dst);
            break;
    }
}
```

在调用初始的输入函数时，要求用一个函数完成层数、起始柱，结束柱三个量的输入，但是函数的返回值只有一个，于是便使用指针的方法，不直接调用而是间接地改变需要记录的量。

由于设置了多个全局性的量，并在一次main循环中对其可能造成了改变，所以在main执行最后需要为其初始化，以便再一次循环时保持其原始情况。

```
/*共用的输入函数*/
int input(char* SRC, char* TMP, char* DST)
{
    int n, i;
    /*层数n*/
    while (1){ ... }
    while (1) {
        cout << "请输入起始柱(A-C): " << endl;
        cin >> *SRC;
        if ((*SRC == 'A' || *SRC == 'a') && (cin.good() == 1)){ ... }
        else if ((*SRC == 'B' || *SRC == 'b') && (cin.good() == 1)){ ... }
        else if ((*SRC == 'C' || *SRC == 'c') && (cin.good() == 1)){ ... }
        else {
            cin.clear();
            cin.ignore(65536, '\n');
        }
    }
    while (1){ ... }
    *TMP = 198 - *SRC - *DST;
    return n;
}
```

```
        cct_cls();
        Count = 0;
        speedchoice = 5;
        for (i = 0; i < 10; i++) {
            A[i] = 0;
            B[i] = 0;
            C[i] = 0;
        }
        a = 0, b = 0, c = 0;
    }
    return 0;
}
```

4. 调试过程碰到的问题

问题一：汉诺塔图形的盘子

首先是不同盘子的宽度和颜色要求不同，但是打印出来后则相对应的宽度与颜色不能变。同时盘子上下移动的动画效果也会有高度不对的情况。

解决方法是将盘子与之相应的编号联系起来，颜色可以直接使用其编号，宽度可以使用其编号的倍数再加减微调，打印的起始位置也需要相应微调。上下移动时循环次数同样要与编号对应联系起来，同时，也要注意使用空格覆盖时要再用同样的方法把柱子补回来，且不要补多。

```
if (src == 'A' && dst == 'B') {
    for (i = a; y2 - i > 1; i++) {
        cct_showstr(x4 - B[b - 1], y2 - i - 1, " ", B[b - 1], COLOR_WHITE, 2 * B[b - 1] + 1, -1);
        cct_showstr(x4 - B[b - 1], y2 - i, " ", COLOR_BLACK, COLOR_WHITE, 2 * B[b - 1] + 1, -1);
        if (y2 - i > 2) {
            cct_showstr(x4, y2 - i, " ", COLOR_HYELLOW, COLOR_WHITE, 1, -1);
        }
        speedcontrol_ex();
    }
    for (i = x4 - B[b - 1]; i < x5 - B[b - 1]; i++) {
        cct_showstr(i + 1, 1, " ", B[b - 1], COLOR_WHITE, 2 * B[b - 1] + 1, -1);
        cct_showstr(i, 1, " ", COLOR_BLACK, COLOR_WHITE, 1, -1);
        speedcontrol_ex();
    }
}
```

问题二：延时设置

程序中的延时设置多且繁琐，同时，菜单项8中需要控制的速度，到菜单项9中形式又不尽相同。

由于延时设置与调用的部分比较多，我的做法仍是单独使用函数；但是针对有些函数对其的使用有一些差异，可以使用额外的参数控制实现不同的效果，而我是直接又对其进行修饰，算是又设了一个基本相似的函数来弥补有些时候的不足。

```
/*过程中的时间间隔函数*/
void speedcontrol()
{
    if (speedchoice == 0) {
        int h;
        while (1) {
            h = _getch();
            if (h == '\r' || h == '\n') {
                break;
            }
            else {
                continue;
            }
        }
    }
    else {
        Sleep(10 * (5 - speedchoice) * (5 - speedchoice) * (5 - speedchoice));
    }
}
```

```
/*过程中的时间间隔函数*/
void speedcontrol() { ... }

/*时间间隔函数改——特化0的情况*/
void speedcontrol_ex()
{
    if (speedchoice != 0) {
        speedcontrol();
    }
    else {
        Sleep(80);
    }
}
```

问题三：数组步数记录

之前的作业中我使用了静态局部变量来控制步数记录，本次不允许，我在函数里设置简单变

量时总是会改变其大小。

注意到本次允许定义的全局变量里包括了步数，在使用了全局变量后可以实现效果，但是在main下一次循环时步数又没有清掉，于是变在每次，main结束时初始化全局变量，保证下一次循环正常进行。

5. 心得体会

5.1. 注意分解问题

面对整体比较复杂的问题时，将其合理化为一些小部分，分为小函数，逐个实现。这样不仅可以使这一个问题的解决结构安排更合理，而且每一个部分函数都可能对与之相关其他的问题起到作用，提升整体效率。

5.2. 定义变量的准确命名

对变量的清晰定义比我之前预想的起到更大的作用。对函数名、变量名取的具体准确能使程序清晰得多，很多时候与其在函数中间加入大量麻烦的注释，可能还不如在定义量的时候用一行简单的注释加上一个准确的名字来的明白。不要为了省事而频繁使用诸如a、b、c的变量，不清晰易混乱。

5.3. 二维数组使用对本题代码的化简

本次比较遗憾的是没有使用二维数组定义汉诺塔，导致讨论的情况会多出不少重复的代码。在我本次作业已经完成了一部分之后，才在课上听到老师更为简练的二维数组的方法，所以没有再做更改。一维数组的讨论确实会有不少赘余部分有待优化。

5.4. 在指定位置打印字符

打印位置。在每个函数的开头建议只定义一个坐标量直接赋值，其余坐标量建议在此坐标之上改动。如此方便整体调整，而不是一盘散沙。

6. 附件：源程序

```
    /*489 专用的界面切换*/
    void screen(int choose, int n, char src, char tmp, char dst)
    {
        int i;
        int dengxian_x = 10, dengxian_y = 12;
        if (choose != 4) {
            dengxian_y += 15;
        }
        cct_cls();
        cout <<"从 " << src << " 移动到 " << dst << ", 共 " << n << " 层";
        if (choose != 9) {
            cout << ", 延时设置为 " << speedchoice << endl;
        }
        cct_gotoxy(dengxian_x, dengxian_y);
        for (i = 0; i < 25; i++) {
            cout << "=";
        }
        printyABC(choose, src, dst);

        cct_gotoxy(0, dengxian_y + 5);
        cout << "初始:  ";
        printxABC();
        if (choose != 4) {
            startplate(choose, n, src, dst);
        }
        cct_setcursor(CURSOR_INVISIBLE);
    }
```

```
    /*唯一的汉诺塔递归函数*/
    void hanoi(int choose, int n, char src, char tmp, char dst)
    {
        if (n == 1) {
            media(choose, n, src, tmp, dst);
        }
        else {
            hanoi(choose, n - 1, src, dst, tmp);
            media(choose, n, src, tmp, dst);
            hanoi(choose, n - 1, tmp, src, dst);
        }
    }
```

```
    /*输出中转的媒介函数*/
    void media(int choose, int n, char src, char tmp, char dst)
    {
        switch (choose) {
            case 1:
                printstep1(n, src, dst);
                break;
            case 2:
                printstep2(n, src, dst);
                break;
            case 3:
                printstep3(choose, n, src, dst);
                break;
            case 4:
                printstep4(choose, n, src, dst);
                break;
            case 8:
                printstep8(choose, n, src, dst);
                break;
        }
    }
```

```
    /*被中转的打印步骤函数*/
    void printstep1(int n, char src, char dst)
    {…}
    void printstep2(int n, char src, char dst)
    {…}
    void printstep3(int choose, int n, char src, char dst)
    {…}
    void printstep4(int choose, int n, char src, char dst)
    {…}
    void printstep8(int choose, int n, char src, char dst)
    {
        if (speedchoice == 0) {
            while (1) {
                int back = _getch();
                if (back == '\n' || back == '\r') {
                    break;
                }
            }
        }
        trans(choose, src, dst);
        cct_gotoxy(0, 32);
        cct_showstr(0, 32, "第", COLOR_BLACK, COLOR_WHITE, 1, -1);
        cout << setw(4) << ++Count << " 步( " << n << "#: " << src << "-->" << dst << ")  ";
        printxABC();
        changexABC(choose, src, dst, 32);
        printyABC(choose, src, dst);
        changeyABC(choose, src, dst, 29);
        movement_up(choose, n, src, dst);
        movement_down(choose, n, src, dst);
        cct_setcursor(CURSOR_INVISIBLE);
    }
```

装 订 线

```
/*横向形象化输出函数*/
void printxABC()
{
    int i;
    cout << "A:";
    if (A[0] > 0) {
        cout << setw(2) << A[0];
    }
    else {
        cout << "  ";
    }
    for (i = 1; i < 10; i++) {
        if (A[i] > 0) {
            cout << setw(2) << A[i];
        }
        else {
            cout << "  ";
        }
    }
    cout << " B:";
    if (B[0] > 0) {…}
    else {…}
    for (i = 1; i < 10; i++) {…}
    cout << " C:";
    if (C[0] > 0) {…}
    else {…}
    for (i = 1; i < 10; i++) {…}
    cout << endl;
}
```

```
/*横向形象化改变函数*/
void changexABC(int choose, char src, char dst, int y)
{
    int x1 = 23, x2 = x1 + 23, x3 = x2 + 23;
    int x4 = 25, x5 = x4 + 23, x6 = x5 + 23;
    if (src == 'A' && dst == 'B') {
        cct_gotoxy(x4 + 2 * a, y);
        cout << "  ";
        cct_gotoxy(x2 + 2 * b, y);
        cout << setw(2) << B[b - 1];
    }
    else if (src == 'A' && dst == 'C') {…}
    else if (src == 'B' && dst == 'A') {…}
    else if (src == 'B' && dst == 'C') {…}
    else if (src == 'C' && dst == 'A') {…}
    else if (src == 'C' && dst == 'B') {…}
}
```

```
/*纵向形象化改变函数*/
void changeyABC(int choose, char src, char dst, int y)
{
    int x1 = 11, x2 = x1 + 10, x3 = x2 + 10;
    if (src == 'A' && dst == 'B') {
        cct_gotoxy(x1, y - a - 3);
        cout << "  ";
        cct_gotoxy(x2, y - b - 2);
        cout << setw(2) << B[b - 1];
    }
    else if (src == 'A' && dst == 'C') {…}
    else if (src == 'B' && dst == 'A') {…}
    else if (src == 'B' && dst == 'C') {…}
    else if (src == 'C' && dst == 'A') {…}
    else if (src == 'C' && dst == 'B') {…}
}
```

```
/*纵向形象化输出函数*/
void printyABC(int choose, char src, char dst)
{
    int i, xa = 11, ya = 11, xb = xa + 10, yb = ya, xc = xb + 10, yc = yb;
    if (choose != 4) {
        ya += 15;
        yb = ya;
        yc = yb;
    }
    cct_gotoxy(xa + 1, ya + 2);
    cout << "A" << setw(10) << "B" << setw(10) << "C";
    for (i = 0; i < 10; i++) {
        cct_gotoxy(xa, ya);
        if (A[i] > 0) {
            cout << setw(2) << A[i];
            ya--;
        }
        else {
            cout << "  ";
            ya--;
        }
    }
    for (i = 0; i < 10; i++) {
        cct_gotoxy(xb, yb);
        if (B[i] > 0) {…}
        else {…}
    for (i = 0; i < 10; i++) {
        cct_gotoxy(xc, yc);
        if (C[i] > 0) {…}
        else {…}
    }
```

装 订 线

装 订 线

```
    /*图形化输出函数-生成柱*/
    void printcolumn(int choose)
    {
        int x1 = 1, x2 = x1 + 32, x3 = x2 + 32;
        int y1 = 15;
        cct_showstr(x1, y1, "  ", COLOR_HYELLOW, COLOR_WHITE, 23, -1);
        cct_showstr(x2, y1, "  ", COLOR_HYELLOW, COLOR_WHITE, 23, -1);
        cct_showstr(x3, y1, "  ", COLOR_HYELLOW, COLOR_WHITE, 23, -1);
        int x4 = 12, x5 = x4 + 32, x6 = x5 + 32;
        int y2 = 14;
        for (y2 = 14; y2 > 2; y2--) {
            cct_showstr(x4, y2, "  ", COLOR_HYELLOW, COLOR_WHITE, 1, -1);
            Sleep(50);
            cct_showstr(x5, y2, "  ", COLOR_HYELLOW, COLOR_WHITE, 1, -1);
            Sleep(50);
            cct_showstr(x6, y2, "  ", COLOR_HYELLOW, COLOR_WHITE, 1, -1);
            cct_setcursor(CURSOR_INVISIBLE);
            Sleep(80);
        }
    }
```

```
    /*图形化输出函数-初始盘*/
    void startplate(int choose, int n, char src, char dst)
    {
        int i;
        if (choose != 8 && choose != 9) {
            cout << "从 " << src << " 移动到 " << dst << ", 共 " << n << " 层" << endl;
        }
        printcolumn(choose);
        int x4 = 12, x5 = x4 + 32, x6 = x5 + 32;
        int y2 = 14;
        if (src == 'A') {
            for (i = 0; i < n; i++) {
                cct_showstr(x4 - A[i], y2, "  ", A[i], COLOR_WHITE, 2 * A[i] + 1, -1);
                y2--;
                Sleep(50);
            }
        }
        else if (src == 'B') {…}
        else if (src == 'C') {…}
        cct_setcursor(CURSOR_INVISIBLE);
    }
```

```
    /*图形化输出函数——一次移动——trans 后下降*/
    void movement_down(int choose, int n, char src, char dst)
    {
        int i;
        int x4 = 12, x5 = x4 + 32, x6 = x5 + 32;
        int y2 = 14;
        if (dst == 'A') {
            for (i = a - 1; y2 - i > 0; i++) {
                cct_showstr(x4 - A[a - 1], i - a + 2, "  ", A[a - 1], COLOR_WHITE, 2 * A[a - 1] + 1, -1);
                if (i != a - 1) {
                    cct_showstr(x4 - A[a - 1], i - a + 1, "  ", COLOR_BLACK, COLOR_WHITE, 2 * A[a - 1] + 1, -1);
                }
                if (i > a + 1) {
                    cct_showstr(x4, i - a + 1, "  ", COLOR_HYELLOW, COLOR_WHITE, 1, -1);
                }
                speedcontrol_ex();
            }
        }
        else if (dst == 'B') {…}
        else if (dst == 'C') {…}
    }
```

```
extern int A[10], B[10], C[10];
extern int a, b, c;
extern int Count;
extern int speedchoice;

int main()
{
    int n, i;
    char src, tmp, dst;

    /* 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
    cct_setconsoleborder(120, 40, 120, 9000);
    while (1) {
        int choose = menu();
        if (choose == 0) {
            break;
        }
        else if (choose == 4) {
            n = input(&src, &tmp, &dst);
            speedchoice = speedchose();
            screen(choose, n, src, tmp, dst);
            hanoi(choose, n, src, tmp, dst);
        }
        else if (choose == 5) { … }
        else if (choose == 6) { … }
        else if (choose == 7) { … }
        else if (choose == 8) { … }
        else if (choose == 9) { … }
        else {
            n = input(&src, &tmp, &dst);
            hanoi(choose, n, src, tmp, dst);
        }
        /*********************************************/
        if (choose == 1 || choose == 2 || choose == 3) {
            cout << "按回车键继续";
        }
        else {
            cct_showstr(0, 38, "按回车键继续", COLOR_BLACK, COLOR_WHITE, 1, -1);
        }
        while (1) {
            int back = _getch();
            if (back == '\n' || back == '\r') {
                break;
            }
        }
        cct_cls();
        Count = 0;
        speedchoice = 5;
        for (i = 0; i < 10; i++) {
            A[i] = 0;
            B[i] = 0;
            C[i] = 0;
        }
        a = 0, b = 0, c = 0;
    }
```

装 订 线

```
/*9专用的函数*/
void freeplay(int choose, int n, char Src, char Dst)
{
    int y = 33;
    char T[2], src, dst;
    int i;
    cct_showstr(0, y, "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) ：", COLOR_BL
    cct_setcursor(CURSOR_VISIBLE_NORMAL);
    for (i = 0; i < 2; i++) {
        T[i] = '0';
    }
    while (1) {
        while (1) {
            cct_showstr(60, y, " ", COLOR_BLACK, COLOR_WHITE, 20, -1);
            cct_gotoxy(60, y);
            for (i = 0; i < 2; i++) {
                cin >> T[i];
                if (T[i] == 32) {
                    i--;
                }
                else if (T[0] == 'q' || T[0] == 'Q') {
                    cct_showstr(0, y + 1, "游戏中止!!!!!", COLOR_BLACK, COLOR_WHITE, 1, -1);
                    break;
                }
            }
            if (T[0] == 'q' || T[0] == 'Q') {
                src = 0;
                dst = 0;
                break;
            }
            if ((T[0] == 'a' || T[0] == 'A')) {
                src = 'A';
                if (T[1] == 'b' || T[1] == 'B') {
                    dst = 'B';
                    if (a == 0) {
                        cct_showstr(0, y + 1, "源柱为空!", COLOR_BLACK, COLOR_WHITE, 1, -1);
                        Sleep(800);
                        cct_showstr(0, y + 1, "          ", COLOR_BLACK, COLOR_WHITE, 1, -1);
                        i--;
                        continue;
                    }
                    else if (b == 0 || A[a - 1] < B[b - 1]) {
                        break;
                    }
                    else {
                        cct_showstr(0, y + 1, "大盘压小盘，非法移动!", COLOR_BLACK, COLOR_WHI
                        Sleep(800);
                        cct_showstr(0, y + 1, "                      ", COLOR_BLACK, COLOR_WHI
                        i--;
                        continue;
                    }
                }
                else if (T[1] == 'c' || T[1] == 'C') { ... }
            }
            else if ((T[0] == 'b' || T[0] == 'B')) { ... }
            else if ((T[0] == 'c' || T[0] == 'C')) { ... }
        }
        if (src == 0 && dst == 0) {
            break;
        }
        trans(choose, src, dst);
        cct_gotoxy(0, 32);
        cct_showstr(0, 32, "第", COLOR_BLACK, COLOR_WHITE, 1, -1);
        cout << setw(4) << ++Count << " 步( " << n << "#: " << src << "-->" << dst << ")  ";
        printxABC();
        changexABC(choose, src, dst, 32);
        printyABC(choose, src, dst);
        changeyABC(choose, src, dst, 29);
        cct_setcursor(CURSOR_INVISIBLE);
        movement_up(choose, n, src, dst);
        movement_down(choose, n, src, dst);
        cct_setcursor(CURSOR_INVISIBLE);
        if ((Dst == 'A' && a == n) || (Dst == 'B' && b == n) || (Dst == 'C' && c == n)) {
            cct_showstr(0, y + 1, "游戏结束!!!!!", COLOR_BLACK, COLOR_WHITE, 1, -1);
            break;
        }
    }
}
```

```
int menu()
{
    int choose;
    cout << "--------------------------------" << endl
        << "1.基本解" << endl
        << "2.基本解(步数记录)" << endl
        << "3.内部数组显示(横向)" << endl
        << "4.内部数组显示(纵向+横向)" << endl
        << "5.图形解-预备-画三个圆柱" << endl
        << "6.图形解-预备-在起始柱上画n个盘子" << endl
        << "7.图形解 - 预备 - 第一次移动" << endl
        << "8.图形解 - 自动移动版本" << endl
        << "9.图形解 - 游戏版" << endl
        << "0.退出" << endl
        << "--------------------------------" << endl
        << "[请选择:] ";
    while (1) {
        choose = _getch();
        if (choose >= 48 && choose <= 57) {
            cout << choose - 48 << endl;
            return (choose - 48);
        }
    }
}
```