**实验报告**

****

报告名称：汉诺塔综合演示

班级：计算机1班

学号：1651574

姓名：贾昊霖

完成日期：2017年12月18日

1. **题目及基本要求**
   1. **题目**

汉诺塔综合演示

* 1. **基本要求**

将之前所有汉诺塔小题集成在一个程序中，用菜单方式进行选择，并加入图形化演示，有许多限制，比如要共用函数、减少但函数的代码量等等..。

1. **整体设计思路**

输入之后进行菜单选择，先初始化函数，然后色织一系列的参数，然后根据不同的选择，从而在一个选择函数中选择不同的解决方案，主体只有一个函数Hanoi，然而，在递归之间反复调用Switchsolution那个函数，尽享二次选择，从而大大减少了代码量..此选择函数相当于控制器，在控制器中控制整个程序流程。

1. **主要功能的实现**

讲真，每个函数我都写有详尽的注释，并且函数以及变量名都起得让读程序者一下就明白，所以我这里不在过多赘述

说几个关键的函数，其1：hanoi主函数，递归思想，也是整个题的主干，其2：设置全局数组以及全局变量，可以存储abc三个基上的盘子，进而可以实时输出所有盘子上的信息，其3：打印彩色盘子需要调用大量的系统函数，开始没发现老师给的头文件，于是自己写，后来..发现了必须全都改过来...

1. **调试过程碰到的问题**

遇到最大的问题就是自己写的程序与标准程序之间的差别，比如哪里没有光标显示，哪里需要把颜色调回黑白..其余没有遇到太多的问题，唯一就是很花时间..断点调试还不是很熟练，不知道vs有没有像Linux下条件断点的那种强大功能..

1. **心得体会**

本次作业其中有一个下午是在上课的时候写的，另外两天晚上从9点写到凌晨1：45左右，然后周一晚上到现在一直完善并写实验报告..总体来说这次作业量很大很足..作为一个完美主义者，总想把事情做的很好，但是做这个花费了确实很多的时间，但是也学到了很多的东西，比如，熟悉了windows.h里边的很多函数，以及增强了写程序的熟练度，如何在短时间内高质高效地完成.

1. **附件：源程序**

/\*1651574 1班 贾昊霖\*/

#include <iostream>

#include <iomanip>

#include <conio.h>

#include <windows.h>

#include <cmath>

#include "cmd\_console\_tools.h"

#define UNCOLOR\_BASE\_X 12

#define UNCOLOR\_PILLAR\_Y 9

#define COLOR\_PILLAR\_Y 12

#define CHOICE4\_X 17

#define CHOICE89\_X 30

#define BASE\_X 16

#define BASE\_Y 1

#define BASE\_LENGTH 23

#define BASE\_HALF 11

#define BASE\_TOP 3

#define BASE\_INTERVAL 9

#define MOVESPEED 5

#define END12\_X 25

#define END12\_Y 60

#define END37\_X 30

#define END37\_Y 100

#define END89\_X 40

#define END89\_Y 100

#define QUIT 0

#define OK 1

#ifdef \_WINGDI\_

#define ERROR -1

#endif

using namespace std;

const int PILLAR\_INTERVAL = BASE\_LENGTH + BASE\_INTERVAL - 1;

const int PILLAR\_START = BASE\_Y + BASE\_HALF;

const int CHOICE8\_BASE = BASE\_X + 12;

const int CHOICE9\_BASE = CHOICE8\_BASE + 7;

const int ColorNumber[] = { 1,9,3,2,11,10,14,12,13,4,5,7,15,8 };//跑马灯颜色顺序

const char tips[] = "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) ：";

const int LEN\_TIPS = strlen(tips);

const HANDLE hout = GetStdHandle(STD\_OUTPUT\_HANDLE);

typedef int Status;

void PrintEasterEgg();

int Plate[3][15];

int p[3], n, step;

/\* 延迟时间设置 \*/

inline void SleepTime(int delay)

{

switch (delay) {

case(0):

while (\_getch() != '\r')

; //按下回车才能继续

break;

case(1):

Sleep(1000);

break;

case(2):

Sleep(500);

break;

case(3):

Sleep(200);

break;

case(4):

Sleep(80);

break;

case(5):

Sleep(20);

break;

default:

Sleep(MOVESPEED);

}

}

/\* 等待换行确认 \*/

void pause()

{

cout << "按回车键继续\n";

while (\_getch() != '\r')

;

}

/\* 初始化栈中元素 \*/

void InitPlates(char Start, char End)

{

p[0] = p[1] = p[2] = 1;

if (Start == 'A') {

for (int i = 1; i <= n; i++)

Plate[0][i] = n - i + 1;

p[0] = n + 1;

}

else if (Start == 'B') {

for (int i = 1; i <= n; i++)

Plate[1][i] = n - i + 1;

p[1] = n + 1;

}

else {

for (int i = 1; i <= n; i++)

Plate[2][i] = n - i + 1;

p[2] = n + 1;

}

}

/\* 操作栈顶指针改变栈中元素 \*/

void MovePlate(char from, char to)

{

int tmp;

switch (from) {

case('A'):

tmp = Plate[0][--p[0]];

break;

case('B'):

tmp = Plate[1][--p[1]];

break;

default:

tmp = Plate[2][--p[2]];

}

switch (to) {

case('A'):

Plate[0][p[0]++] = tmp;

break;

case('B'):

Plate[1][p[1]++] = tmp;

break;

default:

Plate[2][p[2]++] = tmp;

}

}

/\* 打印纵向汉诺塔 \*/

void PrintVertical(int choice)

{

setcolor(hout, COLOR\_BLACK, COLOR\_HWHITE);

for (int j = 0; j < 3; j++) {

for (int i = 10; i >= 1; i--) {

if (choice == 8 || choice == 9)

gotoxy(hout, UNCOLOR\_PILLAR\_Y + 2 + 10 \* j, CHOICE8\_BASE - i);

else

gotoxy(hout, UNCOLOR\_PILLAR\_Y + 2 + 10 \* j, UNCOLOR\_BASE\_X - i);

if (i >= p[j])

cout << ' ';

else

cout << Plate[j][i];

}

}

}

/\* choice == 3 or 4 or 8 or 9 打印横向汉诺塔 \*/

void PrintTransversal(int choice, char from, char to)

{

if (choice == 8 || choice == 9)

gotoxy(hout, 0, CHOICE89\_X);

else if (choice == 4)

gotoxy(hout, 0, CHOICE4\_X);

if (step == 0)

cout << "初始： ";

else {

cout << "第" << setw(4) << step << " 步(";

cout << setw(2) << step << "#: ";

cout << from << "-->" << to << ")";

}

cout << "A:";

for (int i = 1; i <= 10; i++) {

if (i < p[0])

cout << setw(2) << Plate[0][i];

else

cout << " ";

}

cout << " B:";

for (int i = 1; i <= 10; i++) {

if (i < p[1])

cout << setw(2) << Plate[1][i];

else

cout << " ";

}

cout << " C:";

for (int i = 1; i <= 10; i++) {

if (i < p[2])

cout << setw(2) << Plate[2][i];

else

cout << " ";

}

//gotoxy(hout, 0, 17);

putchar('\n');

}

/\* 画彩色的盘子 \*/

void DrawColorPlates()

{

int x, y, pillar, length;

for (int i = 0; i < 3; i++) {

gotoxy(hout, COLOR\_PILLAR\_Y + i \* PILLAR\_INTERVAL, BASE\_X);

x = COLOR\_PILLAR\_Y + i \* PILLAR\_INTERVAL;

y = BASE\_X - 1;

pillar = p[i];

length = --pillar;

for (int i = 0; i < pillar; i++) {

showch(hout, x - length, y - i, ' ', ColorNumber[i], ColorNumber[i], 2 \* length + 1);

length--;

}

}

SleepTime(3);

}

/\* 画柱子 \*/

void DrawColorPillar()

{

for (int i = 0; i < 3; i++)

showch(hout, BASE\_Y + PILLAR\_INTERVAL \* i, BASE\_X, ' ', COLOR\_HYELLOW, COLOR\_HYELLOW, BASE\_LENGTH);

for (int i = 0; i < 3; i++)

for (int j = BASE\_TOP; j < BASE\_X; j++)

showch(hout, PILLAR\_START + i \* PILLAR\_INTERVAL, j, ' ', COLOR\_HYELLOW, COLOR\_HYELLOW, 1);

setcolor(hout, COLOR\_BLACK, COLOR\_HWHITE);

SleepTime(5);

}

/\* choice == 1 or 2 时的输出 \*/

void PrintPrimaryHanoi(int choice, int num, int step, char from, char to)

{

if (choice == 1) {

cout << "#" << num << " " << from << "---->" << to << endl;

}

else {

cout << "第" << setw(4) << step << " 步( ";

cout << setw(2) << num << "#: ";

cout << from << "-->" << to << ")\n";

}

}

/\* 打印初始化元素 \*/

void PrintInitial(int choice, char Start, char End, int delay)

{

setcolor(hout, COLOR\_BLACK, COLOR\_HWHITE);

gotoxy(hout, 0, 0);

cout << "从 " << Start << " 移动到 " << End << "，共 " << p[0] + p[1] + p[2] - 3

<< " 层";

if (choice == 6 || choice == 7)

return;

if (choice != 9)

cout << ",延时设置为 " << delay;

if (choice == 8 || choice == 9)

gotoxy(hout, UNCOLOR\_PILLAR\_Y, CHOICE8\_BASE);

else

gotoxy(hout, UNCOLOR\_PILLAR\_Y, UNCOLOR\_BASE\_X);

cout << "=========================";

if (choice == 8 || choice == 9)

gotoxy(hout, UNCOLOR\_PILLAR\_Y, CHOICE8\_BASE + 1);

else

gotoxy(hout, UNCOLOR\_PILLAR\_Y, UNCOLOR\_BASE\_X + 1);

cout << " A B C";

PrintTransversal(choice, Start, End);

SleepTime(delay);

PrintVertical(choice);

SleepTime(delay);

}

/\* 动画演示盘子移动函数 \*/

void SolveMovement(char from, char to)

{

int start = from - 'A';

int end = to - 'A';

int inc = (from > to) ? -1 : 1;

int PillarStart = p[start];//最上的序号

int PillarEnd = p[end];//最上的序号

int InfoStart = Plate[start][PillarStart - 1];//盘子号

int y = PILLAR\_START + start \* PILLAR\_INTERVAL;

int x = BASE\_X - PillarStart + 1;

int tmp\_color = ColorNumber[n - InfoStart];//对应的颜色

/\* 向上移动 \*/

while (x != 1) {

/\* 删除盘子 \*/

showch(hout, y - InfoStart, x, ' ', COLOR\_BLACK, COLOR\_BLACK, InfoStart);

if (x >= 3)

showch(hout, y, x, ' ', COLOR\_HYELLOW, COLOR\_HYELLOW, 1);

else

showch(hout, y, x, ' ', COLOR\_BLACK, COLOR\_BLACK, 1);

showch(hout, y + 1, x, ' ', COLOR\_BLACK, COLOR\_BLACK, InfoStart);

x--;

SleepTime(5);

/\* 画新盘子 \*/

showch(hout, y - InfoStart, x, ' ', tmp\_color, tmp\_color, 2 \* InfoStart + 1);

SleepTime(5);

}

/\* 向左or右移动 \*/

int Destination\_y = PILLAR\_START + end \* PILLAR\_INTERVAL;

while (y != Destination\_y) {

showch(hout, y - InfoStart, x, ' ', COLOR\_BLACK, COLOR\_BLACK, 2 \* InfoStart + 1);

y += inc;

SleepTime(5);

showch(hout, y - InfoStart, x, ' ', tmp\_color, tmp\_color, 2 \* InfoStart + 1);

SleepTime(5);

}

/\* 向下移动 \*/

int Destination\_x = BASE\_X - PillarEnd;

while (x != Destination\_x) {

/\* 删除盘子 \*/

showch(hout, y - InfoStart, x, ' ', COLOR\_BLACK, COLOR\_BLACK, InfoStart);

if (x >= 3)

showch(hout, y, x, ' ', COLOR\_HYELLOW, COLOR\_HYELLOW, 1);

else

showch(hout, y, x, ' ', COLOR\_BLACK, COLOR\_BLACK, 1);

showch(hout, y + 1, x, ' ', COLOR\_BLACK, COLOR\_BLACK, InfoStart);

x++;

SleepTime(5);

/\* 画新盘子 \*/

showch(hout, y - InfoStart, x, ' ', tmp\_color, tmp\_color, 2 \* InfoStart + 1);

SleepTime(5);

}

}

/\* 递归中根据choice选择解决方案 \*/

void SwitchSolutions(int choice, int num, int delay, char from, char to)

{

switch (choice) {

case(1):case(2):

PrintPrimaryHanoi(choice, num, step, from, to);

break;

case(3):

MovePlate(from, to);

PrintTransversal(choice, from, to);

break;

case(4):

MovePlate(from, to);

PrintTransversal(4, from, to);

SleepTime(delay);

PrintVertical(choice);

SleepTime(delay);

break;

case(7):

if (step >= 2)

return;

SolveMovement(from, to);

break;

case(8):

SolveMovement(from, to);

MovePlate(from, to);

setcolor(hout, COLOR\_BLACK, COLOR\_HWHITE);

PrintTransversal(8, from, to);

SleepTime(delay);

PrintVertical(choice);

SleepTime(delay);

break;

}

}

/\* hanoi主解决函数 \*/

void Hanoi(int num, char from, char to, char by, int choice, int delay)

{

if (num == 1) {

step++;

SwitchSolutions(choice, num, delay, from, to);

return;

}

Hanoi(num - 1, from, by, to, choice, delay);

step++;

SwitchSolutions(choice, num, delay, from, to);

Hanoi(num - 1, by, to, from, choice, delay);

}

/\* 判断字符合法性 \*/

bool JudgeChar(char c)

{

if (c <= 'C' && c >= 'A')

return true;

else if (c <= 'c' && c >= 'a')

return true;

return false;

}

/\* 判断输入的命令是否合法 \*/

Status JudgeValidity(char \*str)

{

int start, end;

int StartPlate, EndPlate;

if (strlen(str) > 2)

return ERROR;

if (strlen(str) == 1 && str[0] == 'Q' || str[0] == 'q') {

gotoxy(hout, 0, CHOICE9\_BASE + 1);

cout << "游戏中止!!!!!";

SleepTime(0);

return QUIT;

}

for (int i = 0; i < 2; i++)

if (!JudgeChar(str[i]))

return ERROR;

if (str[0] > 'C') {

str[0] -= 32;

start = str[0] - 'A';

}

else

start = str[0] - 'A';

if (str[1] > 'C') {

str[1] -= 32;

end = str[1] - 'A';

}

else

end = str[1] - 'A';

if (start == end)

return ERROR;

StartPlate = Plate[start][p[start] - 1];

EndPlate = Plate[end][p[end] - 1];

if (p[start] == 1) {

gotoxy(hout, 0, CHOICE9\_BASE + 1);

cout << "源柱为空!";

SleepTime(1);

showch(hout, 0, CHOICE9\_BASE + 1, ' ', COLOR\_BLACK, COLOR\_BLACK, strlen("源柱为空!"));

return ERROR;

}

if (StartPlate > EndPlate && p[end] > 1) {

gotoxy(hout, 0, CHOICE9\_BASE + 1);

cout << "大盘压小盘，非法移动!";

SleepTime(1);

showch(hout, 0, CHOICE9\_BASE + 1, ' ', COLOR\_BLACK, COLOR\_BLACK, strlen("大盘压小盘，非法移动!"));

return ERROR;

}

return OK;

}

/\* 汉诺塔游戏 \*/

void PlayGame(char from, char to)

{

char command[20];

int flag;

setcolor(hout, COLOR\_BLACK, COLOR\_HWHITE);

gotoxy(hout, 0, CHOICE9\_BASE);

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) ：";

while (true) {

gotoxy(hout, LEN\_TIPS, CHOICE9\_BASE);

setcursor(hout, CURSOR\_VISIBLE\_NORMAL);

cin >> command;

setcursor(hout, CURSOR\_INVISIBLE);

flag = JudgeValidity(command);

if (flag == OK) {

step++;

SolveMovement(command[0], command[1]);

MovePlate(command[0], command[1]);

setcolor(hout, COLOR\_BLACK, COLOR\_HWHITE);

PrintTransversal(8, command[0], command[1]);

SleepTime(4);

PrintVertical(9);

SleepTime(4);

/\*游戏成功\*/

if (p[to - 'A'] == n + 1) {

gotoxy(hout, 0, CHOICE9\_BASE + 1);

cout << "游戏结束!!!!!";

return;

}

}

else if (flag == QUIT)

return;

/\*清理命令\*/

showch(hout, LEN\_TIPS, CHOICE9\_BASE, ' ', COLOR\_BLACK, COLOR\_HWHITE, strlen(command));

}

}

/\* 初始化设置 \*/

void InitialSetting(int choice, int\*delay, char \*Start, char\*End)

{

while (true) {

cout << "请输入汉诺塔层数(1-10):\n";

cin >> n;

if (!cin.good()) {

cin.clear();

cin.ignore(1024, '\n');

continue;

}

if (n > 0 && n < 11)

break;

}

while (true) {

cout << "请输入起始柱(A-C) \n";

cin >> \*Start;

if (!cin.good()) {

cin.clear();

cin.ignore(1024, '\n');

continue;

}

if (JudgeChar(\*Start))

break;

}

while (true) {

cout << "请输入目标柱(A-C) \n";

cin >> \*End;

if (!cin.good()) {

cin.clear();

cin.ignore(1024, '\n');

continue;

}

if (\*End == \*Start || labs(\*End - \*Start) == 32) {

cout << "目标柱(" << End << ")不能与起始柱(" << \*Start << ")相同" << endl;

continue;

}

if (JudgeChar(\*End))

break;

}

if (\*Start > 'C')

\*Start -= 32;

if (\*End > 'C')

\*End -= 32;

/\* 延迟 \*/

if (choice == 4 || choice == 8)

while (true) {

cout << "请输入移动速度(0-5：0-按回车单步演示 1-延时最长 5-延时最短)\n";

cin >> \*delay;

if (!cin.good()) {

cin.clear();

cin.ignore(1024, '\n');

continue;

}

if (\*delay >= 0 && \*delay <= 5)

break;

}

}

/\* 主函数中根据choice选择解决方案 \*/

void Solve(int choice, char Start, char End, int delay)

{

step = 0;

switch (choice) {

case(1):case(2):

break;

case(3):

InitPlates(Start, End);

break;

case(4):

InitPlates(Start, End);

PrintInitial(choice, Start, End, delay);

break;

case(5):

DrawColorPillar();

return; //ATTENTION!!!!!

case(6):

InitPlates(Start, End);

PrintInitial(choice, Start, End, delay);

DrawColorPillar();

DrawColorPlates();

return; //ATTENTION!!!!!

case(7):

InitPlates(Start, End);

PrintInitial(choice, Start, End, delay);

DrawColorPillar();

DrawColorPlates();

break;

case(8):

InitPlates(Start, End);

PrintInitial(choice, Start, End, delay);

DrawColorPillar();

DrawColorPlates();

break;

case(9):

InitPlates(Start, End);

PrintInitial(choice, Start, End, delay);

DrawColorPillar();

DrawColorPlates();

PlayGame(Start, End);

}

Hanoi(n, Start, End, char(3 \* 'B' - Start - End), choice, delay);

}

/\* main \*/

int main()

{

int choice, delay;

char Start, End;

while (true) {

setconsoleborder(hout, END12\_Y, END12\_X);

setcursor(hout, CURSOR\_VISIBLE\_NORMAL);

system("color 0F");

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0-9]";

do {

choice = \_getch();

choice -= '0';

} while (choice < 0 || choice > 9);

if (!choice)

break;

cout << choice << endl;

if (choice != 5)

InitialSetting(choice, &delay, &Start, &End);

else {

delay = 0;

Start = End = 'a';

}

if (choice >= 3 && choice <= 7)

setconsoleborder(hout, END37\_Y, END37\_X);

else if (choice >= 8)

setconsoleborder(hout, END89\_Y, END89\_X);

setcursor(hout, CURSOR\_INVISIBLE);

Solve(choice, Start, End, delay);

setcolor(hout, COLOR\_BLACK, COLOR\_HWHITE);

if (choice == 3)

putchar('\n');

if (choice >= 4 && choice <= 7)

gotoxy(hout, 0, END37\_X - 2);

else if (choice >= 8)

gotoxy(hout, 0, END89\_X - 2);

pause();

system("cls");

}

gotoxy(hout, 0, END12\_X - 1);

setconsoleborder(hout, 75, 50);

system("color F0");

setcolor(hout, COLOR\_HWHITE, COLOR\_BLACK);

PrintEasterEgg();

return 0;

}

/\* Easter Egg \*/

void PrintEasterEgg()

{

printf("::\n :;J7, :, \

::;7:\n ,ivYi, , ;\

LLLFS:\n :iv7Yi :7ri;j5\

PL\n ,:ivYLvr ,ivrrirrY2X,\n\

:;r@Wwz.7r: :ivu@kexianli.\n \

:iL7::,:::iiirii:ii;::::,,irvF7rvvLujL7ur\n \

ri::,:,::i:iiiiiii:i:irrv177JX7rYXqZEkvv17\n \

;i:, , ::::iirrririi:i:::iiir2XXvii;L8OGJr71i\n :\

,, ,,: ,::ir@mingyi.irii:i:::j1jri7ZBOS7ivv,\n ,:\

:, ::rv77iiiriii:iii:i::,rvLq@huhao.Li\n ,, ,, \

,:ir7ir::,:::i;ir:::i:i::rSGGYri712:\n ::: ,v7r:: ::rrv7\

7:, ,, ,:i7rrii:::::, ir7ri7Lri\n , 2OBBOi,iiir;r:: \

,irriiii::,, ,iv7Luur:\n ,, i78MBBi,:,:::,:, :7FSL\

: ,iriii:::i::,,:rLqXv::\n : iuMMP: :,:::,:ii;2GY7OBB0v\

iiii:i:iii:i:::iJqL;::\n , ::::i ,,,,, ::LuBBu BBBBBEri\

i:i:i:i:i:i:i:r77ii\n , : , ,,:::rruBZ1MBBqi, :,,,\

:::,::::::iiriri:\n , ,,,,::::i: @arqiao. \

,:,, ,:::ii;i7:\n :, rjujLYLi ,,:::::,:::::::::,, ,:i,\

:,,,,,::i:iii\n :: BBBBBBBBB0, ,,::: , ,:::::: , ,,\

,, ,,:::::::\n i, , ,8BMMBBBBBBi ,,:,, ,,, , , , , ,\

:,::ii::i::\n : iZMOMOMBBM2::::::::::,,,, ,,,,,,:,,,::\

::i:irr:i:::,\n i ,,:;u0MBMOG1L:::i:::::: ,,,::, ,,, ::::::\

i:i:iirii:i:i:\n : ,iuUuuXUkFu7i:iii:i:::, :,:,: ::::::::i:i:\

::::iirr7iiri::\n : :rk@Yizero.i:::::, ,:ii:::::::i:::::i::,\

::::iirrriiiri::,\n : 5BMBBBBBBSr:,::rv2kuii:::iii::,:i:,,\

, ,,:,:i@petermu.,\n , :r50EZ8MBBBBGOBBBZP7::::i::,:::::,\

: :,:,::i;rrririiii::\n :jujYY7LS0ujJL7r::,::i::,:::::\

:::::::::iirirrrrrrr:ii:\n ,: :@kevensun.:,:,,,::::i:i::\

:::,,::::::iir;ii;7v77;ii;i,\n ,,, ,,:,::::::i:iiiii:\

i::::,, ::::iiiir@xingjief.r;7:i,\n , , ,,,:,,::::::::iiiiii\

iiii:,:,:::::::::iiir;ri7vL77rrirri::\n :,, , ::::::::i:::i\

:::i:i::,,,,,:,::i:i:::iir;@Secbone.ii:::\n");

cout << "总算做完了..累死我了..给您一个微笑..\n";

}