装 订 线

大作业——汉诺塔实验报告

计1 1751151 郭思远

完成日期：12月19日

1. 汉诺塔

输入起始柱，目标住，和圆盘数量（1-10），起始柱上从小到大有n个原盘，需要按大小顺序重新摆放在另一根柱子上。并且规定，大圆盘始终不能压小圆盘，在三根柱子之间一次只能移动一个圆盘，求移动的最少步数。

1.1基本解+基本解（步数记录）

给出最少步数每一步的移动方向和移动盘号，并记录步数。

1.2内部数组显示（横向+纵向）

要求给出移动过程中每根柱子上现有的圆盘数量及编号。显示方式为横式加竖式。

1.3图形解预备

在屏幕上画出三根圆柱，从左到右编号为A、B、C，输入起始柱和圆盘数量（1-10），在起始柱上从小到大画n个盘，盘子颜色各不相同。并画出第一次移动的动画，要求从起始柱上移出，平移后下落到目标柱。

1.4图形解自动移动版

自动完成全部最小步数移动过程，移动方式必须是上移，平移，下移。

1.5游戏版汉诺塔（人工操作移动步骤）

键盘输入两个字母代表本次移动的源和目标，移动时检查合理性（大盘压小盘，源柱为空提示出错，并重输），合理移动记录步数。移动必须上移，平移，下移，待所有按序移动到结束柱则提示“游戏结束”。

2. 整体设计思路

使用三个全局一维数组来显示内部数组和三个全局变量atop,btop,ctop记录数组的栈顶+1。

整个程序的核心是递归函数，递归函数需实现将n-1个盘子移到中间柱，将第n个盘子移到目标柱，再将n-1个盘子移到目标柱。

用三个一维数组模拟进出栈的情况。

在递归函数中调用模拟栈的函数，在模拟栈的函数中加入输出函数，输出内部数组。

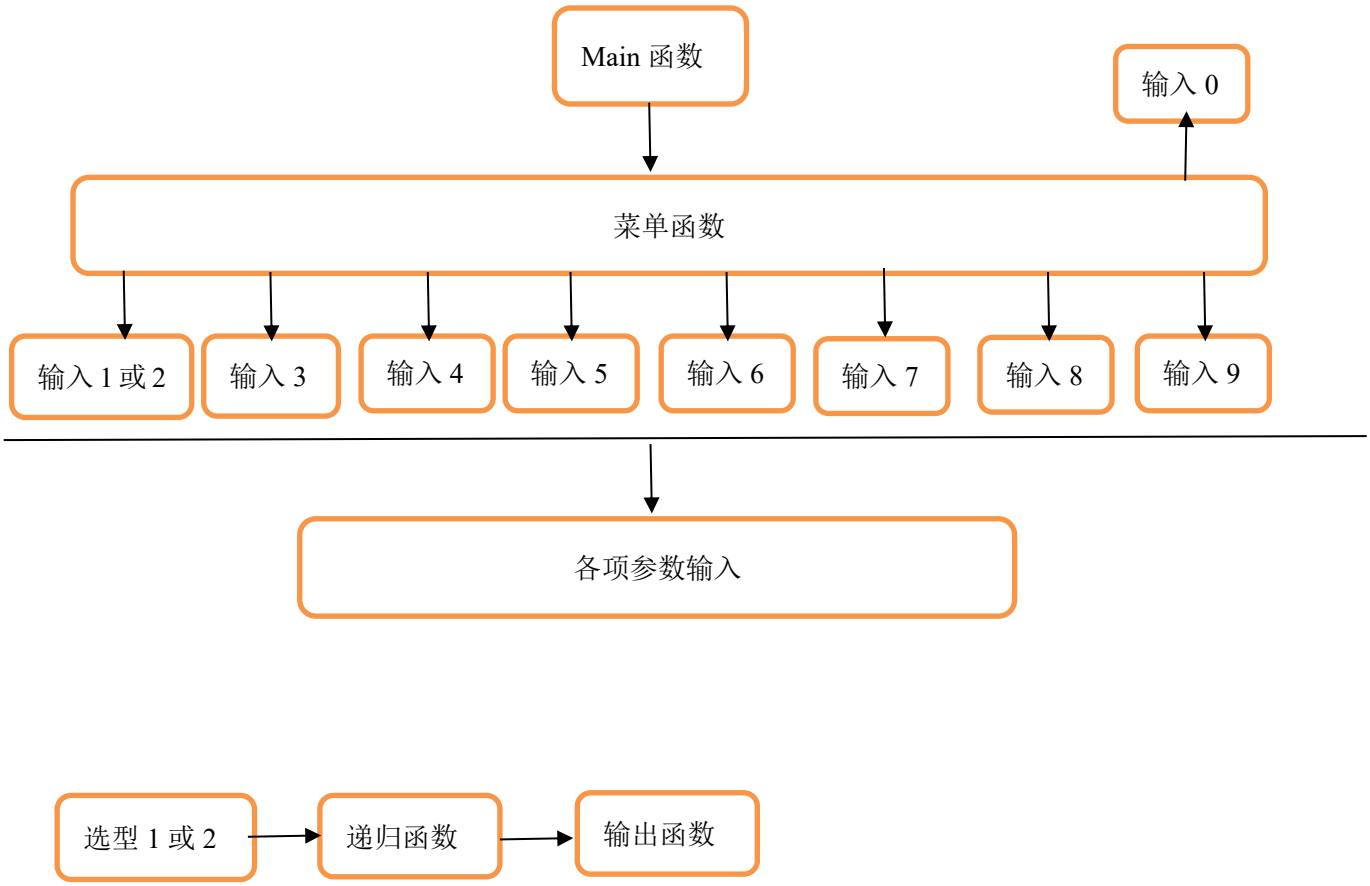
每一个主要功能由一个函数完成，其他函数调用该函数完成一个选项的要求。

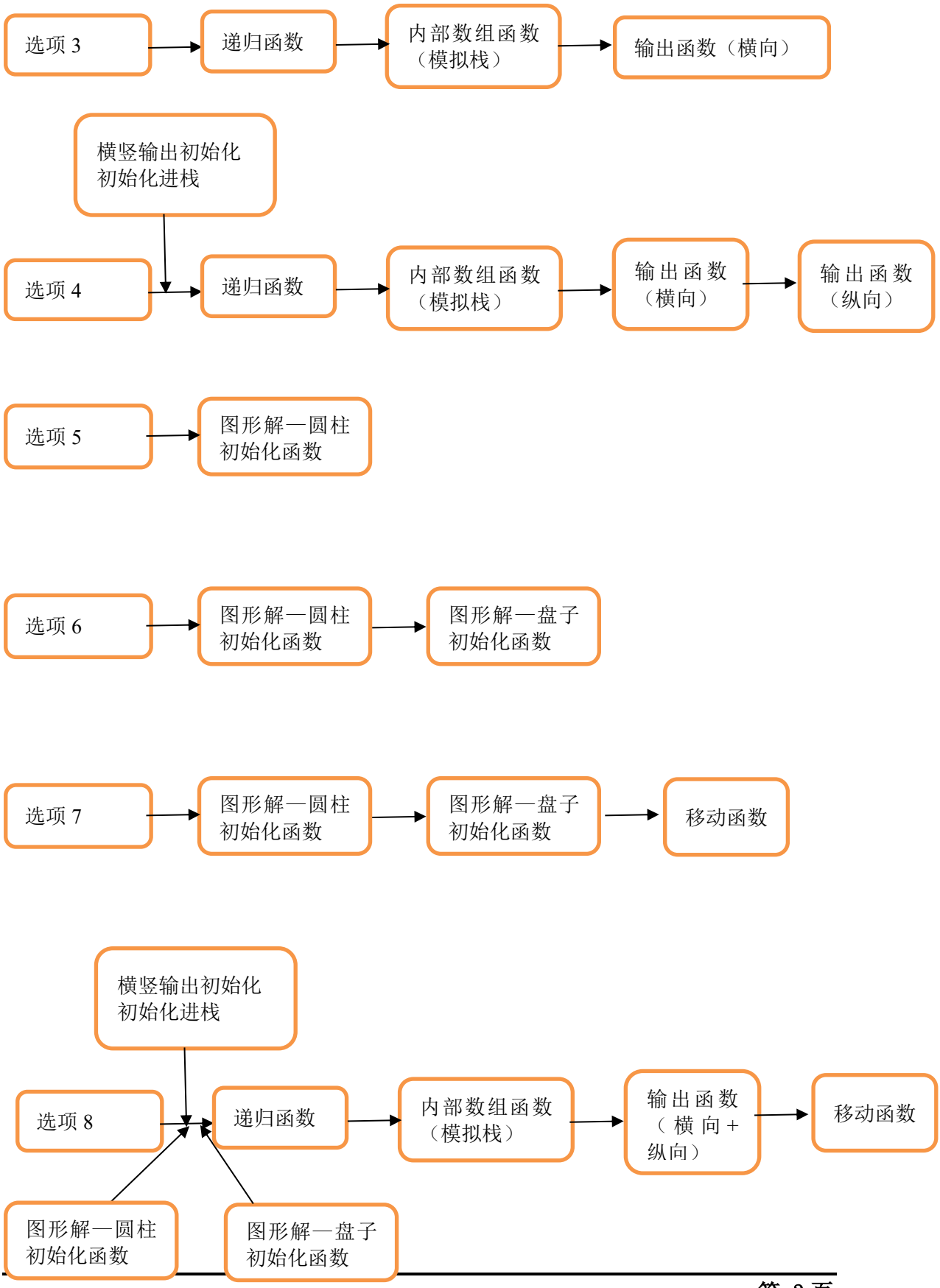
装

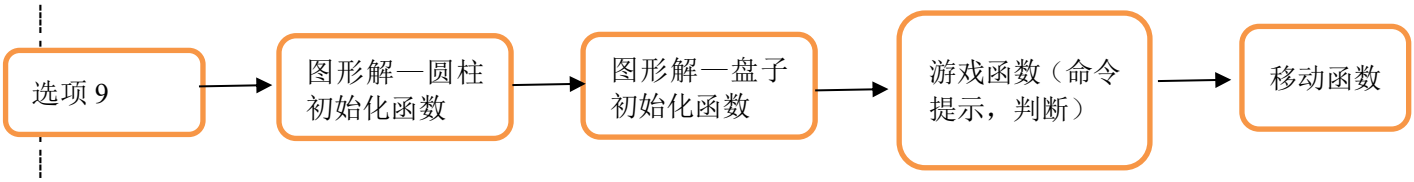
订

线

3. 主要功能的实现




选项 3
递归函数
内部数组函数（模拟栈）
输出函数（横向）
横竖输出初始化
初始化进栈
选项 4
递归函数
内部数组函数（模拟栈）
输出函数（横向）
输出函数（纵向）
选项 5
图形解—圆柱初始化函数
选项 6
图形解—圆柱初始化函数
图形解—盘子初始化函数
选项 7
图形解—圆柱初始化函数
图形解—盘子初始化函数
移动函数
横竖输出初始化
初始化进栈
选项 8
递归函数
内部数组函数（模拟栈）
输出函数（横向+纵向）
移动函数
图形解—圆柱初始化函数
图形解—盘子初始化函数



4. 调试过程碰到的问题

4.1.

显示内部数组时（竖式），在打完数组后会接着打印0出来。一开始想的解决办法是输出一次清屏一次，但是闪得特别厉害而且整体感觉很丑。于是想到从美观角度来说应该只屏蔽零的位置，于是通过每一次从top位开始到10结束输出“ ”进行消除。

4.2.

在图形解移动函数调试的过程中，暴露了很多写第一次移动没考虑到的问题。移动色块颜色混乱，消除函数没有完全消除，错位，盘子不能完全落下去等等问题。一开始想暴力写，定死每个盘子的颜色，大小，后想一想觉得太冗长了。又想到可以通过盘号h这个参数去固定颜色和大小，用表示栈顶的全局变量去控制下落位置，使得代码较为精简。

4.3.

选择一个选项，执行完毕后按回车键返回菜单，再选择下一个选项执行时发现步数记录没有清零。一开始想去掉静态变量，改为传参记录，失败（并且觉得方法不好）。后来又想到可以一开始记录下目标柱，通过表示栈顶的全局变量等于盘子总数判断递归结束，然后把静态变量清零。后来发现可以用外置开关方法，传入一个参数，该参数为零就把这个静态变量清零，修改较少，而且比较精简。

5. 心得体会

5.1. 完成本次作业的心得体会，经验教训

之前没有精简代码使得我非常痛心和后悔，花了不少时间把之前的烂代码改得好看了许多。多了以后每周四晚上把上周作业进行一定修改。

每一个主要功能用一个函数输出，在查找问题时效果很好，改bug的时间明显缩短。并且借此作业学会了之前一直没太弄明白的debug，确实很好用。

在写伪图形界面时，对于整体布局一开始没有考虑周全，导致排版很丑，希望以后吸

取教训。

5.2.在做一些较为复杂的程序时，是分成若干小题好，还是直接一道大题好？

分成若干小题好。可以比较快地上手，难度会因为分解而下降，主观上也更愿意去不断尝试解决问题。 随着问题不断加深，解题思路也有一种连贯性和整体感。

5.3. 前后小题的关联性问题.

总体上注意到了前后小题的关联性。

更好重用代码的方法是，代码按照功能归类成为一个函数模板。通过修改利用模板使得更好重用代码。

5.4.如何更好地利用函数来编写复杂程序.

```
using namespace std;
void CD();//菜单
int get_int(char *prompt, int min_value, int max_value, char *p1, char *p2);//输入
void hengchushihua();//横式初始化
void shuchushihua();//竖式初始化
void CSHzhuzi();//初始化图形柱子
void CSHpanzi(int n, char qsz);//初始化盘子图形
void DG(int n, char one, char two, char three, int sleep, int choice, int t);//递归函数
int hanoidg(char from, char to, int sleep, int n, int choice, int t);//内部数组
void go(char from, char to, int h, int num, int choice);//横向输出
void shu();//纵向输出
void move(int n, char qsz, char mbz, int h, int choice);//移动
void game(int n, char qsz, char mbz, int choice);//游戏版
```

面对复杂程序，先将其分解开来。汉诺塔大作业主要的实现功能分解为菜单、输入、递归、内部数组、横纵输出、移动、游戏版，将这些重要功能列出作为函数，不仅将题目细化拆分使得容易入手，而且使得查找bug，修改程序更加快速、方便。

另外由于初始化需要的次数也比较多，可以拿出来作为一个函数，使得代码精简，整体有一种整洁感。

对于常见的功能和模板要整理成为模板，比如输入函数get_int.在以后面对其他复杂程序时可以套用模板，减少时间消耗。

6.源程序

```
/*1751151 计1 郭思远*/
#define _CRT_SECURE_NO_WARNINGS
#include<iostream>
#include <cstdio>
#include<windows.h>
#include<iomanip>
#include <conio.h>
#include<cmath>
#include "cmd_console_tools.h"
using namespace std;
void CD();//菜单
int get_int(char *prompt, int min_value, int
max_value, char *p1, char *p2);//输入
void hengchushihua();//横式初始化
void shuchushihua();//竖式初始化
void CSHzhuzi();//初始化图形柱子
void CSHpanzi(int n, char qsz);//初始化盘子图形
void DG(int n, char one, char two, char three,
int sleep, int choice, int t);//递归函数
int hanoidg(char from, char to, int sleep, int
n, int choice, int t);//内部数组
void go(char from, char to, int h, int num, int
choice);//横向输出
void shu();//纵向输出
void move(int n, char qsz, char mbz, int h, int
choice);//移动
void game(int n, char qsz, char mbz, int
choice);//游戏版
void setcursor(const HANDLE hout, const int
options);
#define N 10
int a[N], b[N], c[N];
int atop = 0, btop = 0, ctop = 0;
int main()
{
	CD();
	HANDLE hout =
GetStdHandle(STD_OUTPUT_HANDLE);
	gotoxy(hout, 0, 20);
	return 0;
}
int get_int(char *prompt, int min_value, int
max_value, char *p1, char *p2)//输入
{
	const int BASE_X = 10;
	const int BASE_Y = 28;
	HANDLE hout =
GetStdHandle(STD_OUTPUT_HANDLE);
	char a[3];
	int value1;
	char value2;
	if (min_value == 'a')
	{
		while (1)
		{
			gotoxy(hout, BASE_X, BASE_Y +
14);
			cout << "
" << endl;
			gotoxy(hout, BASE_X, BASE_Y +
14);
			cout << prompt;
			cin >> a;
			if (a[0] >= 'a'&&a[0] <=
'z'&&a[1] >= 'a'&&a[1] <= 'z')
			{
				a[0] = a[0] - 32;
				a[1] = a[1] - 32;
			}
			if ((a[0] >= 'A'&&a[0] <=
'C'&&a[1] >= 'A'&&a[1] <= 'C' && (a[0] != a[1]))
|| (a[0] == 'Q'))
			{
				*p1 = a[0];
				*p2 = a[1];
				break;
			}
		}
		return 0;
	}
	if (min_value == 'A')
	{
		while (1)
		{
			cout << prompt << "(" <<
(char)min_value << "-" << (char)max_value << ")"
<< endl;
			cin >> value2;
			if (value2 == 'a' || value2 ==
'b' || value2 == 'c')
			{
				value2 = value2 - 32;
				break;
			}
			else if (value2 == 'A' ||
value2 == 'B' || value2 == 'C')
				break;
		}
		return value2;
	}
	else
	{
		while (1)
		{
			if (max_value == 9)
			{
				gotoxy(hout, 0, 13);
				cout << "
";
				gotoxy(hout, 0, 13);
				cout << prompt;
			}
			else
			{
				cout << prompt << endl;
			}
			cin >> value1;
			if (!cin.good())
			{
				cin.clear();

	cin.ignore(cin.rdbuf()->in_avail(), '\n');
				continue;
			}
			if (value1 >= min_value&&value1
<= max_value)
				break;
		}
		return value1;
	}
}
void CSHzhuzi()//初始化图形柱子
```

```
{
	int i, j;
	HANDLE hout =
GetStdHandle(STD_OUTPUT_HANDLE);
	const int BASE_X = 10;
	const int BASE_Y = 28;
	const int bg_color = COLOR_HYELLOW;
	const int fg_color = COLOR_WHITE;
	const char ch = ' ';
	for (i = 0; i < 3; i++)
	{
		showch(hout, BASE_X + i * 36,
BASE_Y, ch, bg_color, fg_color, 26);
		showch(hout, (BASE_X + 26) + 36 * i,
BASE_Y, ch, COLOR_BLACK, fg_color, 10);
	}
	for (j = 0; j < 15; j++)
	{
		showch(hout, BASE_X + 13, BASE_Y -
j, ch, bg_color, fg_color, 1);
		showch(hout, BASE_X + 13 + 1 * 36,
BASE_Y - j, ch, bg_color, fg_color, 1);
		showch(hout, BASE_X + 13 + 2 * 36,
BASE_Y - j, ch, bg_color, fg_color, 1);
		Sleep(100);
	}
	showch(hout, 100, BASE_Y + 100, ' ',
COLOR_BLACK, COLOR_WHITE, 1);
}
void CD()
{
	int num = 1;
	int choice;
	char gsy;
	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;
	choice = get_int("请选择[0-9]", 0, 9, 0,
0);
	if (choice != 0)
	{
		HANDLE hout =
GetStdHandle(STD_OUTPUT_HANDLE);
		int n, i = 0;
		int sleep;
		char qsz, mbz, zjz;
		const int BASE_X = 10;
		const int BASE_Y = 40;
		if (choice != 5)
		{
			n = get_int("请输入塔的层数(1-
10):", 1, 10, 0, 0);
			qsz = get_int("请输入起始柱",
'A', 'C', 0, 0);
			while (1)
			{
				mbz = get_int("请输入目标
柱", 'A', 'C', 0, 0);
				if (mbz == qsz)
					continue;
				else
					break;
			}
			zjz = 'A' + 'B' + 'C' - mbz -
qsz;

			if (qsz == 'A')
				atop = n;
			if (qsz == 'B')
				btop = n;
			if (qsz == 'C')
				ctop = n;

			for (i = 0; i < N; i++)//初始化
				a[i] = b[i] = c[i] = 0;
			for (i = 0; i < n; i++)//进栈
			{
				if (qsz == 'A')
					a[i] = n - i;
				else if (qsz == 'B')
					b[i] = n - i;
				else if (qsz == 'C')
					c[i] = n - i;
			}
		}
		if (choice == 4 || choice == 8)
		{
			sleep = get_int("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)",
0, 5, 0, 0);
		}
		setcursor(hout, CURSOR_INVISIBLE);
		setconsoleborder(hout, 120, 50);
		if (choice == 4 || choice == 8 ||
choice == 9)
			hengchushihua();
		if (choice == 4 || choice == 8 ||
choice == 9)
			shuchushihua();
		if (choice >= 5)
		{
			CSHzhuzi();
			if (choice == 5)
			{
				gotoxy(hout, 0,36 );
				cout << "按回车键继续" <<
endl;
				gsy = _getch();
				system("cls");
				CD();
			}
		}
		if (choice >= 6)
		{
			CSHpanzi(n, qsz);
			if (choice == 6)
			{
				gotoxy(hout, 0, 36);
				cout << "按回车键继续" <<
endl;
				gsy = _getch();
```

装 订 线

```
                    system("cls");
                    CD();
                }
            }
if (choice == 7)
    move(n, qsz, mbz, 1, choice);
if (choice != 9 && choice != 5 && choice != 6 &&
choice != 7)
{
 if (choice == 4 || choice == 8)
      DG(n, qsz, zjz, mbz, sleep, choice, 1);
else
      DG(n, qsz, zjz, mbz, 6, choice, 1);
if (choice == 4 || choice == 8)
      DG(n, qsz, zjz, mbz, sleep, choice, 0);
else
      DG(n, qsz, zjz, mbz, 6, choice, 0);
if(choice==8||choice==4)
      gotoxy(hout, 0, 45);
cout << "按回车键继续" << endl;
      gsy = _getch();
      system("cls");
      CD();
}
            if (choice == 9)
            {
                  game(n, qsz, mbz, choice);
                  gotoxy(hout, 0, 45);
                  cout << "按回车键继续" << endl;
                  gsy = _getch();
                  system("cls");
                  CD();
            }
      }
}
void CSHpanzi(int n, char qsz)//初始化盘子图形
{
      int i, j = 1;
      HANDLE hout =
GetStdHandle(STD_OUTPUT_HANDLE);
      const int BASE_X = 10;
      const int BASE_Y = 28;
      const int fg_color = COLOR_WHITE;
      const char ch = ' ';
      for (i = 10; i >= 1; i--)
      {
            if (n == i)
            {
                  showch(hout, BASE_X + (12 - n)
+ (qsz - 'A') * 36, BASE_Y - j, ch, 2 + i,
fg_color, 25 - 2 * (11 - n));
                  j++;
                  n--;
            }
      }
      showch(hout, BASE_X, BASE_Y + 40, ' ',
COLOR_BLACK, COLOR_WHITE, 1);
}
void move(int n, char qsz, char mbz, int h, int
choice)//移动
{
      HANDLE hout =
GetStdHandle(STD_OUTPUT_HANDLE);
      const int BASE_X = 10;
      const int BASE_Y = 28;
      const int fg_color = COLOR_HBLUE;
      int y, x, luo, m;
      if (qsz == 'A')
            m = atop;
      if (qsz == 'B')
            m = btop;
      if (qsz == 'C')
            m = ctop;
      if (choice == 7)
      {
            /* 将第一个盘子从下向上移动 */
            for (y = 0; y < 19 - n; y++)
            {
                  showch(hout, BASE_X + 9 + (qsz
- 'A') * 36 + 2, BASE_Y - n - y, ' ', 3,
fg_color, 5);
                  Sleep(100);
                  if (y < 15 - n)
                  {
      showch(hout, BASE_X + 9 + (qsz - 'A') * 36
+ 4, BASE_Y - n - y, ' ', COLOR_HYELLOW,
COLOR_WHITE, 1);
                        showch(hout, BASE_X + 9 +
(qsz - 'A') * 36 + 2, BASE_Y - n - y, ' ',
COLOR_BLACK, COLOR_WHITE, 2);
                        showch(hout, BASE_X + 9 +
(qsz - 'A') * 36 + 5, BASE_Y - n - y, ' ',
COLOR_BLACK, COLOR_WHITE, 2);
                  }
                  if (y >= 15 - n && y < 18 - n)
                        showch(hout, BASE_X + 9 +
(qsz - 'A') * 36 + 2, BASE_Y - n - y, ' ',
COLOR_BLACK, COLOR_WHITE, 5);
            }
            /* 将第一个盘子从当前柱子平移到另一
根柱子 */
for (x = 0; x < 37; x++)
    {
      if (qsz != 'B')
      {
        showch(hout, BASE_X + 9 + (qsz - 'A') *
36 + 2 + x*('A' - qsz + 1), BASE_Y - n - y + 1,
' ', 3, fg_color, 5);
      Sleep(100);
    if (x < 36)
        showch(hout, BASE_X + 9 + (qsz - 'A') *
36 + 2 + x*('A' - qsz + 1), BASE_Y - n - y + 1,
' ', COLOR_BLACK, COLOR_WHITE, 5);
                  }
                  if (qsz == 'B'){
showch(hout, BASE_X + 9 + (qsz - 'A') * 36 + 2 +
x*('A' - qsz), BASE_Y - n - y + 1, ' ', 3,
fg_color, 5);
 Sleep(100);
if (x < 36)
    showch(hout, BASE_X + 9 + (qsz - 'A') * 36 +
2 + x*('A' - qsz), BASE_Y - n - y + 1, ' ',
COLOR_BLACK, COLOR_WHITE, 5);
                  }
            }
            /* 将第一个盘子从当前柱子下落到另一
根柱子 */
for (luo = 0; luo < 18; luo++)
  {
      if (qsz != 'B'){
      showch(hout, BASE_X + 8 + (qsz - 'A') * 36
+ 2 + x*('A' - qsz + 1) + (qsz - 'A'), BASE_Y -
n - y + 1 + luo, ' ', 3, fg_color, 5);
      Sleep(100);
      if (luo < 4)
         showch(hout, BASE_X + 8 + (qsz - 'A') *
```

```
36 + 2 + x*('A' - qsz + 1) + (qsz - 'A'), BASE_Y
- n - y + 1 + luo, ' ', COLOR_BLACK,
COLOR_WHITE, 5);
if (luo >= 4 && luo < 17)
  {
    showch(hout, BASE_X + 10 + 36 + 3, BASE_Y -
n - y + 1 + luo, ' ', COLOR_HYELLOW,
COLOR_WHITE, 1);
  showch(hout, BASE_X + 10 + 36 + 4, BASE_Y - n
- y + 1 + luo, ' ', COLOR_BLACK, COLOR_WHITE,
2);
  showch(hout, BASE_X + 10 + 36 + 1, BASE_Y - n
- y + 1 + luo, ' ', COLOR_BLACK, COLOR_WHITE,
2);
                }
                }
if (qsz == 'B')
{
showch(hout, BASE_X + 8 + (qsz - 'A') * 36 +
x*('A' - qsz) + 4, BASE_Y - n - y + 1 + luo, '
', 3, fg_color, 5);
Sleep(100);
if (luo < 4)
  showch(hout, BASE_X + 8 + (qsz - 'A') * 36 +
x*('A' - qsz) + 4, BASE_Y - n - y + 1 + luo, '
', COLOR_BLACK, COLOR_WHITE, 5);
if (luo >= 4 && luo < 17)
{
  showch(hout, BASE_X + 8 + (qsz - 'A') * 36 + 2
+ x*('A' - qsz) + 4, BASE_Y - n - y + 1 + luo, '
', COLOR_HYELLOW, COLOR_WHITE, 1);
  showch(hout, BASE_X + 8 + (qsz - 'A') * 36 + 2
+ x*('A' - qsz) + 5, BASE_Y - n - y + 1 + luo, '
', COLOR_BLACK, COLOR_WHITE, 2);
  showch(hout, BASE_X + 8 + (qsz - 'A') * 36 + 2
+ x*('A' - qsz) + 2, BASE_Y - n - y + 1 + luo, '
', COLOR_BLACK, COLOR_WHITE, 2);
                }
                }
          }
     }
     if (choice == 8 || choice == 9)
     {
     for (y = m; y < 19 - n; y++)
          {
               showch(hout, BASE_X + 9 + (qsz
- 'A') * 36 + 2 - (h - 1), BASE_Y - y - 1, ' ',
3 + h - 1, fg_color, 5 + 2 * (h - 1));
               if (choice == 8)
                    Sleep(100);
               if (y < 15 - n)
               {
                    showch(hout, BASE_X + 9 +
(qsz - 'A') * 36 + 4, BASE_Y - y - 1, ' ',
COLOR_HYELLOW, COLOR_WHITE, 1);
                    showch(hout, BASE_X + 9 +
(qsz - 'A') * 36 + 2 - (h - 1), BASE_Y - y - 1,
' ', COLOR_BLACK, COLOR_WHITE, 2 + 1 * (h - 1));
                    showch(hout, BASE_X + 9 +
(qsz - 'A') * 36 + 5, BASE_Y - y - 1, ' ',
COLOR_BLACK, COLOR_WHITE, 2 + 1 * (h - 1));
               }
               if (y >= 15 - n&& y < 19 - n)
                    showch(hout, BASE_X + 9 +
(qsz - 'A') * 36 + 2 - (h - 1), BASE_Y - y - 1,
' ', COLOR_BLACK, COLOR_WHITE, 5 + 2 * (h - 1));
          }
          for (x = 0; x < 37 * abs((int)(mbz -
qsz)); x++)
{
if (qsz != 'B')
{
showch(hout, BASE_X + 9 + (qsz - 'A') * 36 + 2 +
x*('A' - qsz + 1), BASE_Y - n - y + 1, ' ', 3 +
(h - 1), fg_color, 5 + 1 * (h - 1));
if (choice == 8)
  Sleep(100);
if (x < 37 * abs((int)(mbz - qsz)))
    showch(hout, BASE_X + 9 + (qsz - 'A') * 36 +
2 + x*('A' - qsz + 1), BASE_Y - n - y + 1, ' ',
COLOR_BLACK, COLOR_WHITE, 5 + 1 * (h - 1));
               }
               if (qsz == 'B')
               {
                    showch(hout, BASE_X + 9 +
(qsz - 'A') * 36 + 2 + x*(mbz - qsz), BASE_Y - n
- y + 1, ' ', 3 + (h - 1), fg_color, 5 + 2 * (h
- 1));
                    if (choice == 8)
                         Sleep(100);
                    if (x < 37)
                         showch(hout, BASE_X
+ 9 + (qsz - 'A') * 36 + 2 + x*(mbz - qsz),
BASE_Y - n - y + 1, ' ', COLOR_BLACK,
COLOR_WHITE, 5 + 2 * (h - 1));
               }
          }
          /* 将第一个盘子从当前柱子下落到另一
根柱子 */
          if (qsz == 'A')
          {
               if (mbz == 'B')
               {
                    for (luo = 0; luo < 18 -
btop + 1; luo++)
                    {
                         showch(hout, BASE_X
+ 8 + (qsz - 'A') * 36 + 1 + x*('A' - qsz + 1) +
('C' - mbz) - (h - 1), BASE_Y - n - y + 1 + luo,
' ', 3 + (h - 1), fg_color, 5 + 2 * (h - 1));
                         if (choice == 8)
                              Sleep(100);
                         if (luo < 4)
                              showch(hout,
BASE_X + 8 + (qsz - 'A') * 36 + 1 + x*('A' - qsz
+ 1) + ('C' - mbz) - (h - 1), BASE_Y - n - y + 1
+ luo, ' ', COLOR_BLACK, COLOR_WHITE, 5 + 2 * (h
- 1));
                         if (luo >= 4 && luo
< 18 - btop)
                         {
                              showch(hout,
BASE_X + 9 + (qsz - 'A') * 36 + x*('A' - qsz +
1) + 2 + ('C' - mbz), BASE_Y - n - y + 1 + luo,
' ', COLOR_HYELLOW, COLOR_WHITE, 1);
                              showch(hout,
BASE_X + 9 + (qsz - 'A') * 36 + x*('A' - qsz +
1) + 3 + ('C' - mbz), BASE_Y - n - y + 1 + luo,
' ', COLOR_BLACK, COLOR_WHITE, 2 + 1 * (h - 1));
                              showch(hout,
BASE_X + 9 + (qsz - 'A') * 36 + x*('A' - qsz +
1) + ('C' - mbz) - (h - 1), BASE_Y - n - y + 1 +
luo, ' ', COLOR_BLACK, COLOR_WHITE, 2 + 1 * (h -
1));
                         }
                    }
```

装 订 线

```
                }
                else
                {
                    for (luo = 0; luo < 18 - ctop + 1; luo++)
                    {
                        showch(hout, BASE_X + 8 + (qsz - 'A') * 36 + 1 + x*('A' - qsz + 1) + ('C' - mbz) - (h - 1), BASE_Y - n - y + 1 + luo, ' ', 3 + (h - 1), fg_color, 5 + 2 * (h - 1));
                        if (choice == 8)
                            Sleep(100);
                        if (luo < 4)
                            showch(hout, BASE_X + 8 + (qsz - 'A') * 36 + 1 + x*('A' - qsz + 1) + ('C' - mbz) - (h - 1), BASE_Y - n - y + 1 + luo, ' ', COLOR_BLACK, COLOR_WHITE, 5 + 2 * (h - 1));
                        if (luo >= 4 && luo < 18 - ctop)
                        {
                            showch(hout, BASE_X + 9 + (qsz - 'A') * 36 + x*('A' - qsz + 1) + 2 + ('C' - mbz), BASE_Y - n - y + 1 + luo, ' ', COLOR_HYELLOW, COLOR_WHITE, 1);
                            showch(hout, BASE_X + 9 + (qsz - 'A') * 36 + x*('A' - qsz + 1) + 3 + ('C' - mbz), BASE_Y - n - y + 1 + luo, ' ', COLOR_BLACK, COLOR_WHITE, 2 + 1 * (h - 1));
                            showch(hout, BASE_X + 9 + (qsz - 'A') * 36 + x*('A' - qsz + 1) + ('C' - mbz) - (h - 1), BASE_Y - n - y + 1 + luo, ' ', COLOR_BLACK, COLOR_WHITE, 2 + 1 * (h - 1));
                        }
                    }
                }
            }
            if (qsz == 'C')
            {
                if (mbz == 'A')
                {
                    for (luo = 0; luo < 18 - atop + 1; luo++)
                    {
                        showch(hout, BASE_X + 8 + (qsz - 'A') * 36 + 5 + x*('A' - qsz + 1) - (h - 1) + ('A' - mbz), BASE_Y - n - y + 1 + luo, ' ', 3 + (h - 1), fg_color, 5 + 2 * (h - 1));
                        if (choice == 8)
                            Sleep(100);
                        if (luo < 4)
                            showch(hout, BASE_X + 8 + (qsz - 'A') * 36 + 5 + x*('A' - qsz + 1) - (h - 1) + ('A' - mbz), BASE_Y - n - y + 1 + luo, ' ', COLOR_BLACK, COLOR_WHITE, 5 + 2 * (h - 1));
                        if (luo >= 4 && luo < 18 - atop)
                        {
                            showch(hout, BASE_X + 9 + (qsz - 'A') * 36 + x*('A' - qsz + 1) + 6 + ('A' - mbz), BASE_Y - n - y + 1 + luo, ' ', COLOR_HYELLOW, COLOR_WHITE, 1);
                            showch(hout, BASE_X + 9 + (qsz - 'A') * 36 + x*('A' - qsz + 1) + 7 + ('A' - mbz), BASE_Y - n - y + 1 + luo, ' ', COLOR_BLACK, COLOR_WHITE, 2 + 1 * (h - 1));
                            showch(hout, BASE_X + 9 + (qsz - 'A') * 36 + x*('A' - qsz + 1) + 4 - (h - 1) + ('A' - mbz), BASE_Y - n - y + 1 + luo, ' ', COLOR_BLACK, COLOR_WHITE, 2 + 1 * (h - 1));
                        }
                    }
                }
                else
                {
                    for (luo = 0; luo < 18 - btop + 1; luo++)
                    {
                        showch(hout, BASE_X + 8 + (qsz - 'A') * 36 + 5 + x*('A' - qsz + 1) + ('A' - mbz) - (h - 1), BASE_Y - n - y + 1 + luo, ' ', 3 + (h - 1), fg_color, 5 + 2 * (h - 1));
                        if (choice == 8)
                            Sleep(100);
                        if (luo < 4)
                            showch(hout, BASE_X + 8 + (qsz - 'A') * 36 + 5 + x*('A' - qsz + 1) + ('A' - mbz) - (h - 1), BASE_Y - n - y + 1 + luo, ' ', COLOR_BLACK, COLOR_WHITE, 5 + 2 * (h - 1));
                        if (luo >= 4 && luo < 18 - btop)
                        {
                            showch(hout, BASE_X + 9 + (qsz - 'A') * 36 + x*('A' - qsz + 1) + 6 + ('A' - mbz), BASE_Y - n - y + 1 + luo, ' ', COLOR_HYELLOW, COLOR_WHITE, 1);
                            showch(hout, BASE_X + 9 + (qsz - 'A') * 36 + x*('A' - qsz + 1) + 7 + ('A' - mbz), BASE_Y - n - y + 1 + luo, ' ', COLOR_BLACK, COLOR_WHITE, 2 + 1 * (h - 1));
                            showch(hout, BASE_X + 9 + (qsz - 'A') * 36 + x*('A' - qsz + 1) + 4 - (h - 1) + ('A' - mbz), BASE_Y - n - y + 1 + luo, ' ', COLOR_BLACK, COLOR_WHITE, 2 + 1 * (h - 1));
                        }
                    }
                }
            }
            if (qsz == 'B')
            {
if (mbz == 'A')
{
    for (luo = 0; luo < 18 - atop + 1; luo++)
    {
     showch(hout, BASE_X + 8 + (qsz - 'A') * 36 + x*(mbz - qsz) + 4 - (mbz - 'A') - (h - 1), BASE_Y - n - y + 1 + luo, ' ', 3 + (h - 1), fg_color, 5 + 2 * (h - 1));
      if (choice == 8)
          Sleep(100);
      if (luo < 4)
        showch(hout, BASE_X + 8 + (qsz - 'A') * 36 + x*(mbz - qsz) + 4 - (mbz - 'A') - (h - 1), BASE_Y - n - y + 1 + luo, ' ', COLOR_BLACK, COLOR_WHITE, 5 + 2 * (h - 1));
      if (luo >= 4 && luo < 18 - atop)
      {
       showch(hout, BASE_X + 8 + (qsz - 'A') * 36 + 2 + x*(mbz - qsz) + 4 - (mbz - 'A'), BASE_Y - n - y + 1 + luo, ' ', COLOR_HYELLOW, COLOR_WHITE, 1);
```

装 订 线

```
      showch(hout, BASE_X + 8 + (qsz - 'A') * 36
+ 2 + x*(mbz - qsz) + 5 - (mbz - 'A'), BASE_Y -
n - y + 1 + luo, ' ', COLOR_BLACK, COLOR_WHITE,
2 + 1 * (h - 1));
      showch(hout, BASE_X + 8 + (qsz - 'A') * 36
+ 2 + x*(mbz - qsz) + 2 - (h - 1) - (mbz - 'A'),
BASE_Y - n - y + 1 + luo, ' ', COLOR_BLACK,
COLOR_WHITE, 2 + 1 * (h - 1));
                              }
                        }
                  }
else
    for (luo = 0; luo < 18 - ctop + 1; luo++)
          {
           showch(hout, BASE_X + 8 + (qsz -
'A') * 36 + x*(mbz - qsz) + 4 - (mbz - 'A') - (h
- 1), BASE_Y - n - y + 1 + luo, ' ', 3 + (h -
1), fg_color, 5 + 2 * (h - 1));
          if (choice == 8)
            Sleep(100);
          if (luo < 4)
          showch(hout, BASE_X + 8 + (qsz -
'A') * 36 + x*(mbz - qsz) + 4 - (mbz - 'A') - (h
- 1), BASE_Y - n - y + 1 + luo, ' ',
COLOR_BLACK, COLOR_WHITE, 5 + 2 * (h - 1));
         if (luo >= 4 && luo < 18 - ctop)
          {
            showch(hout, BASE_X + 8 + (qsz -
'A') * 36 + 2 + x*(mbz - qsz) + 4 - (mbz - 'A'),
BASE_Y - n - y + 1 + luo, ' ', COLOR_HYELLOW,
COLOR_WHITE, 1);
          showch(hout, BASE_X + 8 + (qsz -
'A') * 36 + 2 + x*(mbz - qsz) + 5 - (mbz - 'A'),
BASE_Y - n - y + 1 + luo, ' ', COLOR_BLACK,
COLOR_WHITE, 2 + 1 * (h - 1));
showch(hout, BASE_X + 8 + (qsz - 'A') * 36 + 2 +
x*(mbz - qsz) + 2 - (h - 1) - (mbz - 'A'),
BASE_Y - n - y + 1 + luo, ' ', COLOR_BLACK,
COLOR_WHITE, 2 + 1 * (h - 1));
                              }
                        }
          }
     }
     showch(hout, BASE_X, BASE_Y + 40, ' ',
COLOR_BLACK, COLOR_WHITE, 1);
}
void hengchushihua()
{
     const int BASE_X = 10;
     const int BASE_Y = 40;
     int i, j;
     HANDLE hout =
GetStdHandle(STD_OUTPUT_HANDLE); //取标准输出设
备对应的句柄

     gotoxy(hout, BASE_X, BASE_Y + 2);
     cout << "初始:";//输出初始行
     cout << "A: ";
     for (i = 0, j = 0; i < N; i++)
     {
          if (a[i] == 0)
          {
               for (j; j < N - i; j++)
                    cout << setw(2) << " ";
               break;
          }
          else
               cout << setw(1) << a[i] << " ";
     }
     cout << setiosflags(ios::right) << setw(2)
<< "B: ";
     for (i = 0, j = 0; i < N; i++)
     {
          if (b[i] == 0)
          {
               for (j; j < N - i; j++)
                    cout << setw(2) << " ";
               break;
          }
          else
               cout << setw(1) << b[i] << " ";
     }
     cout << setiosflags(ios::right) << setw(2)
<< "C: ";
     for (i = 0; i < N; i++)
     {
          if (c[i] == 0)
          {
               for (j; j < N - i; j++)
                    cout << setw(2) << " ";
               break;
          }
          else
               cout << setw(1) << c[i] << " ";
     }
     cout << endl;

}
void shuchushihua()
{
     const int BASE_X = 10;
     const int BASE_Y = 40;
     int y, k;
     HANDLE hout =
GetStdHandle(STD_OUTPUT_HANDLE);
     gotoxy(hout, BASE_X, BASE_Y);
     cout << "A";
     gotoxy(hout, BASE_X + 1 * 10, BASE_Y);
     cout << "B";
     gotoxy(hout, BASE_X + 2 * 10, BASE_Y);
     cout << "C";
     gotoxy(hout, BASE_X - 10, BASE_Y - 1);
     cout <<
"=======================================" <<
endl;
     for (k = 0; k < N; k++)
     {
          gotoxy(hout, BASE_X, BASE_Y - k -
2);

          if (k >= 0 && k <= atop - 1)
               cout << a[k];
          if (k >= atop && k <= 10)
               cout << "  ";
     }
     for (k = 0; k < N; k++)
     {
          gotoxy(hout, BASE_X + 10, BASE_Y - k
- 2);

          if (k >= 0 && k <= btop - 1)
               cout << b[k];
          if (k >= btop&& k <= 10)
               cout << "  ";
     }
     for (y = 19, k = 0; y > 9 && k < N; y--,
```

```
k++)
      {
            gotoxy(hout, BASE_X + 20, BASE_Y - k
- 2);

            if (k >= 0 && k <= ctop - 1)
                  cout << c[k];
            if (k >= ctop && k <= 10)
                  cout << "  ";
      }
}
void shu()//纵向输出
{
      const int BASE_X = 10;
      const int BASE_Y = 40;
      HANDLE hout =
GetStdHandle(STD_OUTPUT_HANDLE);
      int y, k;
      for (k = 0; k < N; k++)
      {
      gotoxy(hout, BASE_X, BASE_Y - k - 2);
            if (k >= 0 && k <= atop - 1)
                  cout << a[k];
            if (k >= atop && k <= 10)
                  cout << "  ";
      }
      for (k = 0; k < N; k++)
      {
      gotoxy(hout, BASE_X + 10, BASE_Y - k - 2)
            if (k >= 0 && k <= btop - 1)
                  cout << b[k];
            if (k >= btop&& k <= 10)
                  cout << "  ";
      }
      for (y = 19, k = 0; y > 9 && k < N; y--,
k++)
      {
      gotoxy(hout, BASE_X + 20, BASE_Y - k - 2);
            if (k >= 0 && k <= ctop - 1)
                  cout << c[k];
            if (k >= ctop && k <= 10)
                  cout << "  ";
      }
}
int hanoidg(char from, char to, int sleep, int
n, int choice, int t)//内部数组
{
      static int num = 1;
      if (t == 0)
            num = 1;
      if (t == 1)
      {
            int h;//盘号
            int gsy;
            const int BASE_X = 10;
            const int BASE_Y = 40;
            HANDLE hout =
GetStdHandle(STD_OUTPUT_HANDLE);
            if (from == 'A'&&to == 'C')
            {
                  h = c[ctop++] = a[--atop];
                  a[atop] = 0;
                  if (sleep == 0)
                        gsy = _getch();
                  else
                        Sleep((6 - sleep) * 100);
                  if (choice >= 4)
                        gotoxy(hout, BASE_X,
BASE_Y + 2);
                  if (choice == 3)
                        gotoxy(hout, 0, num - 1);
                  go(from, to, h, num, choice);
                  if (choice >= 4)
                        shu();
                  move(n, from, to, h, choice);
                  num++;
            }
            if (from == 'A'&&to == 'B')
            {
                  h = b[btop++] = a[--atop];
                  a[atop] = 0;
                  if (sleep == 0)
                        gsy = _getch();
                  else
                        Sleep((6 - sleep) * 100);
                  if (choice >= 4)
                        gotoxy(hout, BASE_X,
BASE_Y + 2);
                  if (choice == 3)
                        gotoxy(hout, 0, num - 1);
                  go(from, to, h, num, choice);
                  if (choice >= 4)
                        shu();
                  move(n, from, to, h, choice);
                  num++;
            }
            if (from == 'B'&&to == 'C')
            {
                  h = c[ctop++] = b[--btop];
                  b[btop] = 0;
                  if (sleep == 0)
                        gsy = _getch();
                  else
                        Sleep((6 - sleep) * 100);
                  if (choice >= 4)
                        gotoxy(hout, BASE_X,
BASE_Y + 2);
                  if (choice == 3)
                        gotoxy(hout, 0, num - 1);
                  go(from, to, h, num, choice);
                  if (choice >= 4)
                        shu();
                  move(n, from, to, h, choice);
                  num++;
            }
            if (from == 'B'&&to == 'A')
            {
                  h = a[atop++] = b[--btop];
                  b[btop] = 0;
                  if (sleep == 0)
                        gsy = _getch();
                  else
                        Sleep((6 - sleep) * 100);
                  if (choice >= 4)
                        gotoxy(hout, BASE_X,
BASE_Y + 2);
                  if (choice == 3)
                        gotoxy(hout, 0, num - 1);
                  go(from, to, h, num, choice);
                  if (choice >= 4)
                        shu();
                  move(n, from, to, h, choice);
                  num++;
            }
            if (from == 'C'&&to == 'A')
            {
```

```
                h = a[atop++] = c[--ctop];
                c[ctop] = 0;
                if (sleep == 0)
                     gsy = _getch();
                else
                     Sleep((6 - sleep) * 100);
                if (choice >= 4)
                     gotoxy(hout, BASE_X,
BASE_Y + 2);
                if (choice == 3)
                     gotoxy(hout, 0, num - 1);
                go(from, to, h, num, choice);
                if (choice >= 4)
                     shu();
                move(n, from, to, h, choice);
                num++;
           }
           if (from == 'C'&&to == 'B')
           {
                h = b[btop++] = c[--ctop];
                c[ctop] = 0;
                if (sleep == 0)
                     gsy = _getch();
                else
                     Sleep((6 - sleep) * 100);
                if (choice >= 4)
                     gotoxy(hout, BASE_X,
BASE_Y + 2);
                if (choice == 3)
                     gotoxy(hout, 0, num - 1);
                go(from, to, h, num, choice);
                if (choice >= 4)
                     shu();
                move(n, from, to, h, choice);
                num++;

           }
     }
     return 0;
}
void DG(int n, char one, char two, char three,
int sleep, int choice, int t)//递归函数
{
     char jl = three;
     if (n == 1)
          hanoidg(one, three, sleep, n,
choice, t);
else{
  DG(n - 1, one, three, two, sleep, choice, t);
  hanoidg(one, three, sleep, n, choice, t);
  DG(n - 1, two, one, three, sleep, choice, t);
     }
}
void go(char from, char to, int h, int num, int
choice)//横向输出
{
     HANDLE hout =
GetStdHandle(STD_OUTPUT_HANDLE);
     int i, j;
     if (choice == 1)
     {
          gotoxy(hout, 0, num - 1);
          cout << h << "#: " << from << "-->"
<< to << endl;
     }
     if (choice == 2)
     {
          gotoxy(hout, 0, num - 1);
          cout << "第" <<
setiosflags(ios::right) << setw(4) << num << "
步" << "(" << setiosflags(ios::right) << setw(2)
<< h << "#: " << from << "-->" << to << ") " <<
endl;
     }
     if (choice >= 3)
     {

          cout << "第" <<
setiosflags(ios::right) << setw(4) << num << "
步" << "(" << setiosflags(ios::right) << setw(2)
<< h << "#: " << from << "-->" << to << ") ";
          cout << "A: ";
          for (i = 0, j = 0; i < N; i++)
          {
               if (a[i] == 0)
               {
                    for (j; j < N - i; j++)
                         cout << setw(2) << "
";
                    break;
               }
               else
                    cout << setw(1) << a[i]
<< " ";
          }
          cout << setiosflags(ios::right) <<
setw(2) << "B: ";
          for (i = 0, j = 0; i < N; i++)
          {
            if (b[i] == 0)
            {
              for (j; j < N - i; j++)
                cout << setw(2) << " ";
                    break;
                    }
               else
                cout << setw(1) << b[i] << " ";
                    }
cout << setiosflags(ios::right) << setw(2) << "C: ";
          for (i = 0; i < N; i++)
          {
            if (c[i] == 0)
               {
               for (j; j < N - i; j++)
                    cout << setw(2) << " ";
                         break;
                    }
               else
                    cout << setw(1) << c[i] << " ";
               }
               cout << endl;
          }
     }
     void game(int n, char qsz, char mbz, int choice)
     {
          static int num = 1, h;
          while (1)
          {
               const int BASE_X = 10;
               const int BASE_Y = 28;
               HANDLE hout =
     GetStdHandle(STD_OUTPUT_HANDLE);
               gotoxy(hout, BASE_X, BASE_Y + 14);
               cout << "
     " << endl;
               gotoxy(hout, BASE_X, BASE_Y + 14);
```

装 订 线

```
            get_int("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）:", 'a', 'c',
&qsz, &mbz);
            if (qsz == 'q' || qsz == 'Q')
            {
                  gotoxy(hout, BASE_X + 4, BASE_Y
+ 15);
                  cout << "游戏中止!!!! " <<
endl;
                  break;
            }
            int m, j, k;
            if (mbz == 'A')
            {
                  m = atop;
                  k = a[atop - 1];
            }
            if (mbz == 'B')
            {
                  m = btop;
                  k = b[btop - 1];
            }
            if (mbz == 'C')
            {
                  m = ctop;
                  k = c[ctop - 1];
            }
            if (qsz == 'A')
            {
                  j = a[atop - 1];
            }
            if (qsz == 'B')
            {
                  j = b[btop - 1];
            }
            if (qsz == 'C')
            {
                  j = c[ctop - 1];
            }
            if (j == 0)
            {
                  gotoxy(hout, BASE_X + 4, BASE_Y
+ 15);
                  cout << "源柱为空！" << endl;
                  Sleep(300);
                  gotoxy(hout, BASE_X + 4, BASE_Y
+ 15);
                  cout << "          " << endl;
                  continue;
            }
            if (k != 0)
                  if (j > k)
{
      gotoxy(hout, BASE_X + 4, BASE_Y + 15);
      cout << "大盘压小盘，非法移动！" << endl;
      Sleep(300);
      gotoxy(hout, BASE_X + 4, BASE_Y + 15);
                        cout << "
" << endl;
                        continue;
                  }
            if (qsz == 'A'&&mbz == 'C')
            {
                  h = c[ctop++] = a[--atop];
                  a[atop] = 0;
                  gotoxy(hout, BASE_X, BASE_Y +
16);
                  go(qsz, mbz, h, num, choice);
                  shu();
                  num++;
            }

            if (qsz == 'A'&&mbz == 'B')
            {
                  h = b[btop++] = a[--atop];
                  a[atop] = 0;
                  gotoxy(hout, BASE_X, BASE_Y +
16);
                  go(qsz, mbz, h, num, choice);
                  shu();
                  num++;
            }
            if (qsz == 'B'&&mbz == 'C')
            {
                  h = c[ctop++] = b[--btop];
                  b[btop] = 0;
                  gotoxy(hout, BASE_X, BASE_Y +
16);
                  go(qsz, mbz, h, num, choice);
                  shu();
                  num++;
            }
            if (qsz == 'B'&&mbz == 'A')
            {
                  h = a[atop++] = b[--btop];
                  b[btop] = 0;
                  gotoxy(hout, BASE_X, BASE_Y +
16);
                  go(qsz, mbz, h, num, choice);
                  shu();
                  num++;

            }
            if (qsz == 'C'&&mbz == 'A')
            {
                  h = a[atop++] = c[--ctop];
                  c[ctop] = 0;
                  gotoxy(hout, BASE_X, BASE_Y +
16);
                  go(qsz, mbz, h, num, choice);
                  shu();
                  num++;
            }
            if (qsz == 'C'&&mbz == 'B')
            {
                  h = b[btop++] = c[--ctop];
                  c[ctop] = 0;
                  gotoxy(hout, BASE_X, BASE_Y +
16);
                  go(qsz, mbz, h, num, choice);
                  shu();
                  num++;
            }

            move(n, qsz, mbz, h, choice);
            if (m + 1 == n)
            {
                  gotoxy(hout, BASE_X + 4, BASE_Y
+ 15);
                  cout << "游戏结束!!!!" << endl;
                  break;
            }
      }
```

装

订

线