同济大学

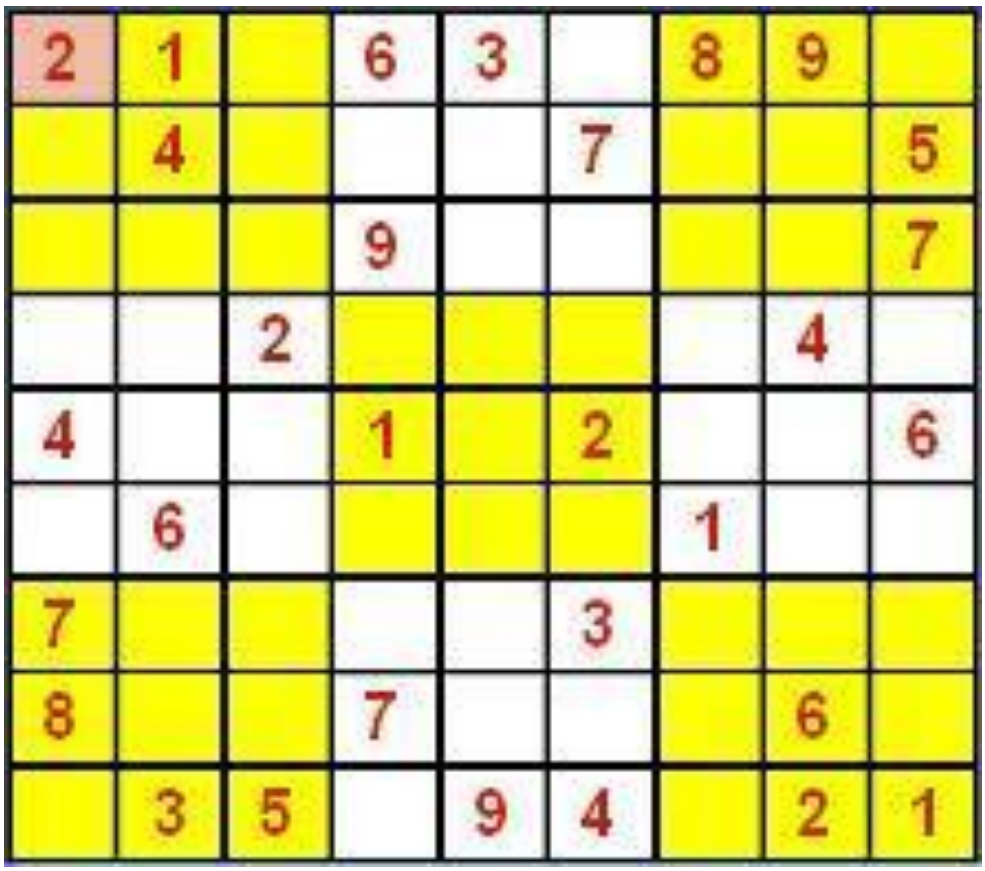
实验报告

装

订

线

数独游戏及数独的自动求解



3班

1553449

王志业

2017.4.17

装 订 线

1. 数独游戏及数独的自动求解

1.1. 题目背景

数独游戏（日语：数独 すうどく）是一种源自18世纪末的瑞士的游戏，后在美国发展、并在日本得以发扬光大的数学智力拼图游戏。拼图是九宫格（即3格宽×3格高）的正方形状，每一格又细分为一个九宫格。在每一个小九宫格中，分别填上1至9的数字，让整个大九宫格每一列、每一行的数字都不重复。 数独的玩法逻辑简单，数字排列方式千变万化。不少教育者认为数独是锻炼脑筋的好方法。

1.2. 基本要求描述

子题目1：输入数独题目文件读取初始数据，判断该文件数据是否合法。有冲突的地方要给出相应提示（用不同颜色标识错误，已有固定数字，未填的数字），输入指定格式填写到制定位置。完成游戏后给出成功的提示信息。

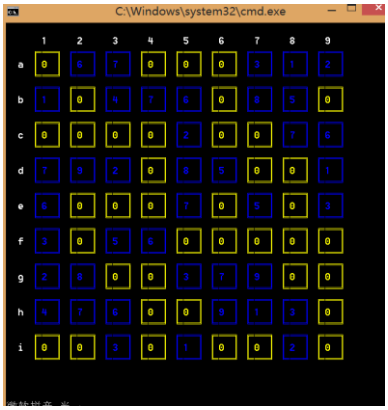
子题目2：获取当前文件下所有以sudoku开头的文件，并模拟列表框，用上下箭头移动回车选择所需要的文件

子题目3：图形界面下的自动求解

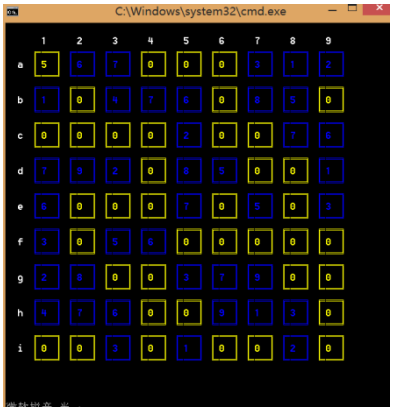
2. 整体设计思路

以子题目3自动求解为准的设计思路:首先观察老师所给的demo程序，判断大致为递归尝试法，即从第一个为0的数字开始，依次尝试1-9中的每一个数字，若某个数字符合当前9*9矩阵的数独规则要求，则进入到下一个为0的数字，尝试1-9，某个数字符合后再进入下一个为0的数字尝试。若某为0数字已经尝试过1-9均不满足要求，则将该位置置回0，然后去找上一个位置，从上一个位置的当前数字再继续尝试。如果第一个为0数字尝试到9仍然不满足后面的要求。那么该数独无解。所以先在已定义好的结构体中再添加一项int num_try，代表我正在尝试的数字。定义函数find搜索为0的位置并填入尝试的数字。定义函数conflict寻找在填入的位置上是否产生了冲突，若没有冲突则继续find。有冲突则返回上一级函数继续尝试。

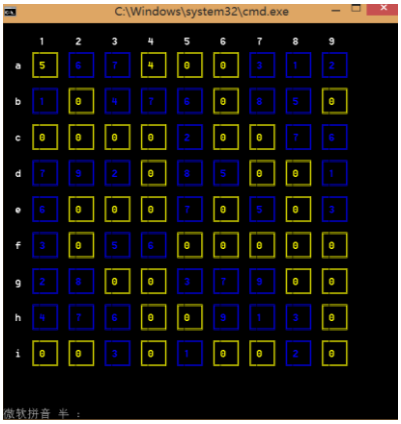
以sudoku-2为例展示设计思路：



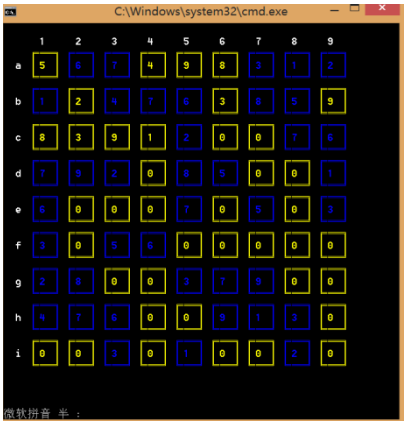
初始状态



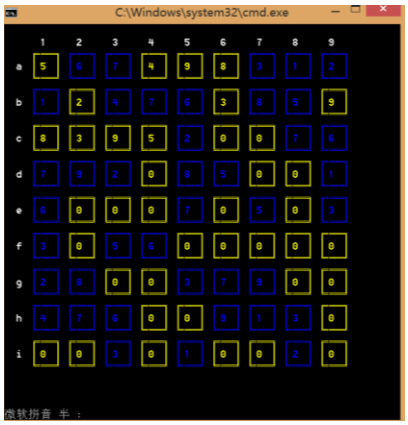
尝试位置a1，从1开始，到5才发现没有冲突，继续搜索下一个为0位置并尝试



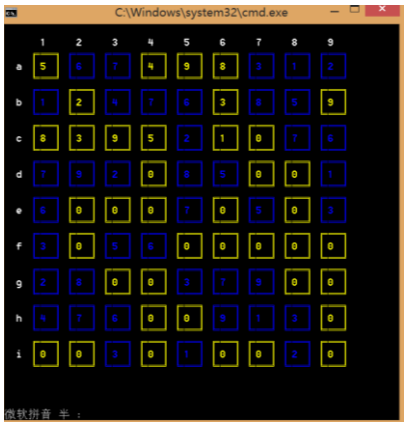
尝试位置a4，到4发现没有冲突，继续搜索下一个为0位置并尝试



尝试到c6发现在这种情况下1-9均不成立，返回c4继续寻找c4的下一个数字

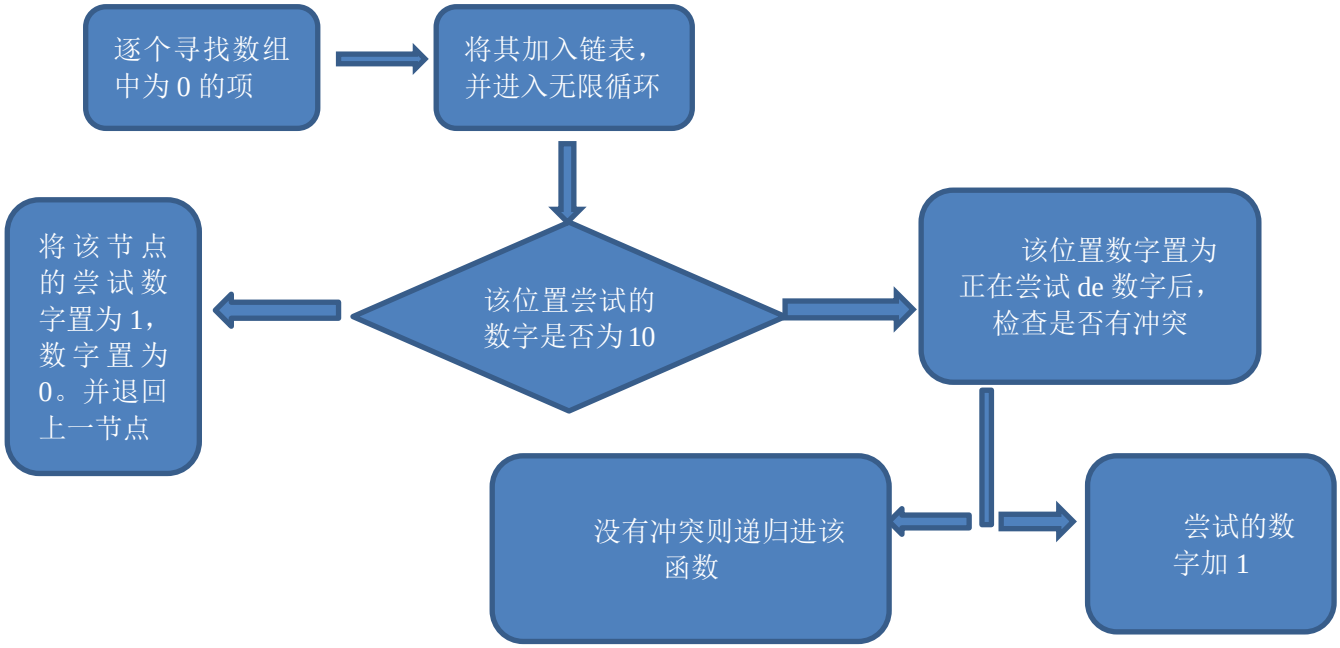


尝试至5满足当前要求再进行下一个位置的尝试



开始下一位置尝试如此反复循环直至全部填好或得出无解的结论

3. 主要功能的实现



```
        while (1)
        {
            if ((*p)->try_num != 10)
            {
                (*p)->num = (*p)->try_num;
                mark(sdkarray, (*p)->x, (*p)->y);
                if (conflict1(sdkarray, i, j) != 0)
                {
                    (*p)->try_num++;
                    continue;
                }
                mark_back(sdkarray, (*p)->x, (*p)->y);
                find(sdkarray, p, q, head);
            }
            else
            {
                if ((*p)->lnext == NULL)
                {
                    setcolor(hout, COLOR_BLACK, COLOR_HWHITE);
                    gotoxy(hout, 2, 30);
                    cout << "wrong!!!";
                    exit(-1);
                }
                else
                {
                    (*p)->try_num = 1;
                    (*p)->num = 0;
                    mark_back(sdkarray, (*p)->x, (*p)->y);
                    *p = (*p)->lnext;
                    *q = *p;
                    (*p)->try_num++;
                }
                return;
            }
        }
    }
}
```

主循环代码如上，当所有的空格都被填满则不能再进入循环，函数结束

4. 调试过程碰到的问题

4.1更改来源于网络的获取当前文件夹下所有某名称某类型文件的过程

```
//获取当前文件下所有sudoku开头文件的名称信息并返回二维字符串指针
char** EnumFiles(int *count)
{
    WIN32_FIND_DATA FindFileData;
    HANDLE hFind;
    char result[MAX_RESULT][MAX_PATH];
    char **returnresult;
    char pattern[MAX_PATH];
    int i = 0, j;

    // 开始查找
    _getcwd(pattern, MAX_RESULT);

    strcat(pattern, "\\sudoku*.*");
    hFind = FindFirstFile(pattern, &FindFileData);

    if (hFind == INVALID_HANDLE_VALUE)
    {
        *count = 0;
        return NULL;
    }
    else
    {
        do
        {
            strcpy(result[i++], FindFileData.cFileName);
        } while (FindNextFile(hFind, &FindFileData) != 0);
    }

    // 查找结束
    FindClose(hFind);

    // 复制到结果中
    returnresult = (char **)calloc(i, sizeof(char *));

    for (j = 0; j < i; j++)
    {
        returnresult[j] = (char *)calloc(MAX_PATH, sizeof(char));
        strcpy(returnresult[j], result[j]);
    }

    *count = i;
    return returnresult;
}
```

代码如上，通过查阅资料可知各部分含义：_getcwd取当前文件的路径到pattern，后面的\\sudoku*.*意思为找到文件名称中部分含sudoku的文件，不限类型，如果想要限制类型可将*改为“.dat”、“.txt”等。

4.2cmd图形界面的构建心得

比如在这道题目中要求对已有数字，后来填写的数字和出错的数字用不同颜色标记。我原来的方法是将该处数字字符化然后用showstr输入这个字符。在本次的编码作业中突然发现可以用更为简单的setcolor先取好颜色然后直接cout输出当前的值。在构建分割线时也有很多窍门，循环最好是与数组的行数和列数一致，方便以后的取鼠标或者键盘控制上下左右的操作。

5. 心得体会

5.1通过完成本次作业，更深入的体会到程序设计的学习主要依靠搜寻资料和自学。比如本次的从当前文件夹下找到所有以sudoku开头的文件，需要自己去查找资料。网上的这种代码数不胜数很多拿过来编译可能由于编译器版本原因不通过，这时候就考察自己的问题解决能力和迁移运用能力。拿过来的东西没办法直接用，要自己去分析每个和每个都是干嘛用的然后自己去修改使其满足自己的需求。

5.2第二次大作业在函数设计上比第一次大作业有了很大的进步，只有一个函数（conflict）用了重复的代码。能更好的利用代码首先要做到有全局观念，重在提前设计。在下手去敲代码之前要对整个题目有一个大致的概念并有一个对代码合理的规划，大概需要用到什么功能的函数，函数的接口参数和函数返回的参数。有整个程序运行的调用函数的流程图，然后根据流程图去编写代码。

6. 附件：源程序

自动求解算法

```
void find(sdk sdkarray[][9], sdk **p, sdk **q,
sdk **head)
{
const HANDLE hout =
GetStdHandle(STD_OUTPUT_HANDLE);
int i, j;
for (i = 0; i < 9; i++)
for (j = 0; j < 9; j++)
{
if (sdkarray[i][j].num == 0)
{
                *p = &sdkarray[i][j];
if (*head == NULL)
*head = *p;
else
{
(*p)->lnext = *q;
(*q)->rnext = *p;
}
*q = *p;
while (1)
{
if ((*p)->try_num != 10)
{
                        (*p)->num =
(*p)->try_num;
mark(sdkarray, (*p)->x, (*p)->y);
if (conflict1(sdkarray, i, j) != 0)
{
(*p)->try_num++;
continue;
}
mark_back(sdkarray, (*p)->x, (*p)->y);
find(sdkarray, p, q, head);
}
else
{
if ((*p)->lnext == NULL)
{
setcolor(hout, COLOR_BLACK, COLOR_HWHITE);
gotoxy(hout, 2, 30);
cout << "wrong!!!";
exit(-1);
}
else
{
(*p)->try_num = 1;
(*p)->num = 0;
mark_back(sdkarray, (*p)->x, (*p)->y);
*p = (*p)->lnext;
*q = *p;
(*p)->try_num++;
return;
}
}
}
}
}
cout << endl << "congratulations!!!";


}
```

多小题公用函数：

1. 文件读取

```
void read_file(sdk sdkarray[][9], char *s)
{
ifstream fin;
int i, j;
fin.open(s, ios::in | ios::binary);
if (fin.is_open() == 0)
{
cout << "open failed";
exit(-1);
}
for (i = 0; i < 9; i++)
{
for (j = 0; j < 9; j++)
{
fin >> sdkarray[i][j].num;
sdkarray[i][j].x = i;
sdkarray[i][j].y = j;
if (sdkarray[i][j].num < 0 ||
sdkarray[i][j].num>9 || fin.fail() == 1)
{
cout << "文件内容不合法";
fin.close();
exit(-1);
}
if (sdkarray[i][j].num>0)
sdkarray[i][j].p = '1';
}

}
fin.close();
```

2. 构建伪图形界面显示当前文件夹中开头为 sudoku 的文件

```
char* display(char **result, const int
count)
{
const HANDLE hout =
GetStdHandle(STD_OUTPUT_HANDLE);
int i, j, address = 1, array_add = 0;
int width = 0, length = 3;
```

装 订 线

```
setcursor(hout, CURSOR_INVISIBLE);
for (i = 0; i < count; i++)
if (strlen(result[i])>unsigned(width))
width = strlen(result[i]);
char *s;//为空行
s = new(nothrow) char[width + 1];
for (i = 0; i < width; i++)
s[i] = ' ';
s[width] = '\0';
showstr(hout, 10, 10, "┌", COLOR_BLACK,
COLOR_HWHITE);
for (i = 2; i <= width; i += 2)
showstr(hout, 10 + i, 10, "—", COLOR_BLACK,
COLOR_HWHITE);
showstr(hout, 10 + i, 10, "┐", COLOR_BLACK,
COLOR_HWHITE);
for (j = 1; j <= length; j++)
showstr(hout, 10 + i, 10 + j, "|",
COLOR_BLACK, COLOR_HWHITE);
showstr(hout, 10 + i, 10 + j, "┘",
COLOR_BLACK, COLOR_HWHITE);
for (; i >= 1; i -= 2)
showstr(hout, 8 + i, 10 + j, "—", COLOR_BLACK,
COLOR_HWHITE);
showstr(hout, 10, 10 + j, "└", COLOR_BLACK,
COLOR_HWHITE);
for (; j > 1; j--)
showstr(hout, 10, 9 + j, "|", COLOR_BLACK,
COLOR_HWHITE);
showstr(hout, 12, 10 + address,
result[array_add], COLOR_WHITE,
COLOR_HWHITE);
showstr(hout, 12, 12, result[array_add + 1],
COLOR_BLACK, COLOR_HWHITE);
showstr(hout, 12, 13, result[array_add + 2],
COLOR_BLACK, COLOR_HWHITE);
while (1)
{
switch (_getch())
{
case 13:
{
gotoxy(hout, 20, 20);
setcolor(hout, COLOR_BLACK, COLOR_HWHITE);
delete s;
cout << result[array_add];
return result[array_add];
}
continue;
case 72://如果读取到上箭头
{

if (array_add > 0)
{
if (address > 1)
{
showstr(hout, 12, 10 + address, s,
COLOR_BLACK, COLOR_HWHITE);
showstr(hout, 12, 10 + address,
result[array_add], COLOR_BLACK,
COLOR_HWHITE);
array_add--;
address--;
}
else
{
array_add--;
showstr(hout, 12, 12, s, COLOR_BLACK,
COLOR_HWHITE);
showstr(hout, 12, 13, s, COLOR_BLACK,
COLOR_HWHITE);
showstr(hout, 12, 12, result[array_add + 1],
COLOR_BLACK, COLOR_HWHITE);
showstr(hout, 12, 13, result[array_add + 2],
COLOR_BLACK, COLOR_HWHITE);

}
showstr(hout, 12, 10 + address, s,
COLOR_WHITE, COLOR_HWHITE);
showstr(hout, 12, 10 + address,
result[array_add], COLOR_WHITE,
COLOR_HWHITE);


}
}
continue;
case 80://如果读取到下键头
{
if (array_add < count - 1)
{

if (address < 3)
{
showstr(hout, 12, 10 + address, s,
COLOR_BLACK, COLOR_HWHITE);
showstr(hout, 12, 10 + address,
result[array_add], COLOR_BLACK,
COLOR_HWHITE);
array_add++;
address++;
}
else
{
array_add++;
showstr(hout, 12, 11, s, COLOR_BLACK,
COLOR_HWHITE);
showstr(hout, 12, 12, s, COLOR_BLACK,
```

```
COLOR_HWHITE);
showstr(hout, 12, 11, result[array_add - 2],
COLOR_BLACK, COLOR_HWHITE);
showstr(hout, 12, 12, result[array_add - 1],
COLOR_BLACK, COLOR_HWHITE);
}
showstr(hout, 12, 10 + address, s,
COLOR_WHITE, COLOR_HWHITE);
showstr(hout, 12, 10 + address,
result[array_add], COLOR_WHITE,
COLOR_HWHITE);
}
}
continue;
default:
continue;

}
}
```

3. }（来源于网络资料）获取当前文件夹下某名称，某类型的文件

```
char** EnumFiles(int *count)
{
WIN32_FIND_DATA FindFileData;
HANDLE hFind;
char result[MAX_RESULT][MAX_PATH];
char **returnresult;
char pattern[MAX_PATH];
int i = 0, j;

// 开始查找
_getcwd(pattern, MAX_RESULT);

strcat(pattern, "\\sudoku*.*");
hFind = FindFirstFile(pattern,
&FindFileData);

if (hFind == INVALID_HANDLE_VALUE)
{
*count = 0;
return NULL;
}
else
{
do
{
strcpy(result[i++],
FindFileData.cFileName);
} while (FindNextFile(hFind,
&FindFileData) != 0);
}

// 查找结束
FindClose(hFind);

// 复制到结果中
returnresult = (char **)calloc(i,
sizeof(char *));

for (j = 0; j < i; j++)
{
returnresult[j] = (char *)calloc(MAX_PATH,
sizeof(char));
strcpy(returnresult[j], result[j]);
}

*count = i;
return returnresult;
```

4. }将某节点加入到链表中（反复使用）

```
void chain(sdk sdkarray[][9], int row, int
line, sdk **p, sdk **q, sdk **head)
{
*p = &sdkarray[row][line];

if ((*head) == NULL)
{
*head = *p;

}
else
{
(*p)->lnext = *q;

(*q)->rnext = *p;
}

*q = *p;
```

5. }将某节点从链表中删除（反复使用）

```
void back(sdk **p, sdk **q)
{
(*p)->num = 0;
*p = (*q)->lnext;
*q = *p;
```

6. 加分题

```
void keyboard(sdk sdkarray[][9], sdk **p,
sdk **q, sdk **head)
{
const HANDLE hout =
GetStdHandle(STD_OUTPUT_HANDLE);
int i = 0, j = 0;//记录当前的位置
mark(sdkarray, i, j);
while (1)
{
switch (_getch())
{
```

```
case 72:
{
if (i > 0)
{
i--;
mark(sdkarray, i, j);
mark_back(sdkarray, i + 1, j);
}
continue;
}

case 80:
{
if (i < 8)
{
i++;
mark(sdkarray, i, j);
mark_back(sdkarray, i - 1, j);
}
continue;
}
case 75:
{
if (j > 0)
{
j--;
mark(sdkarray, i, j);
mark_back(sdkarray, i, j + 1);
}
continue;
}
case 77:
{
if (j < 8)
{
j++;
mark(sdkarray, i, j);
mark_back(sdkarray, i, j - 1);
}
continue;
}
case 13:
{
if (sdkarray[i][j].p == '0')
{
chain(sdkarray, i, j, p, q, head);
int num;
while (1)
{
setcolor(hout, COLOR_BLACK, COLOR_HWHITE);
gotoxy(hout, 2, 30);
cout << "请输入当前位置要更改的数字";
num = _getch() - '0';
gotoxy(hout, 2, 30);
cout << "                              ";
if (num > 0 && num < 10)
{
sdkarray[i][j].num = num;
mark(sdkarray, i, j);
break;
}
gotoxy(hout, 2, 30);
cout << "输入有误请重新输入";

cout << "                              ";
}
}
else
{
setcolor(hout, COLOR_BLACK, COLOR_HWHITE);
gotoxy(hout, 2, 30);
cout << "该位置数字不可更改";
cout << "                              ";
}
continue;
}
case 8:
{
if ((*p)->lnext == NULL)
cout << "已经退回第一步";
else
{
back(p, q);
mark_back(sdkarray, i, j);
mark(sdkarray, (*p)->x, (*p)->y);
i = (*p)->x;
j = (*p)->y;
}
}
continue;
default:
continue;
}
}
}
```

装 订 线