八数码问题

1. 八数码问题：在3\*3的方格棋盘上，摆放1到8这八个数，有一个方格是空的，其初始状态如图，要求对空格执行空格左移、空格右移、空格上移和空格下移这四个操作使得从初始状态到目标状态。

|  |  |  |
| --- | --- | --- |
| 2 | 5 | 4 |
| 3 |  | 7 |
| 1 | 8 | 6 |

初始状态

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 8 |  | 4 |
| 7 | 6 | 5 |

目标状态

#include<stdio.h>

#include<conio.h>

int n,m;

typedef struct Node{

char matrix[10];/\*存储矩阵\*/

char operate;/\*存储不可以进行的操作，L代表不能左移R代表不能右移U代表不能上移D

代表不能下移\*/

char extend;/\*是否可以扩展，Y代表可以，N代表不可以\*/

int father;/\*指向产生 自身的父结点\*/

}Node;

char start[ 10]={" "};/\*此处没有必要初始化\*/

char end[10]={"1238 4765"};/\*此处没有必要初始化\*/

Node base[4000];

int result[ 100];/\*存放结果的base数组下标号，逆序存放\*/

int match()/\*判断是否为目标\*/

{

int i;

for(i=0;i<9;i++){

if(base[n-1].matrix[i]!=end[i])

{return 0;}}

return 1;}

void show()/\*显示矩阵的内容\*/

{

int i=1;

while(m>=0){

int mm=result[m];

//clrscr();

printf("\n\n\n 状态方格\t\t 步骤 %d",i);

printf("\n\n\n\n\n\t\t\t%c\t%c\t%c\t%c\n", base[mm].matrix[0],base[mm].matrix[1],base[mm].matrix[2]);

printf("\n\n\t\t\t%c\t%c\t%c\n",base[mm].matrix[3],base[mm].matrix[4],base[mm].matrix[5]);

printf("\n\n\t\t\t%c\t%c\t%c\n",base[mm].matrix[6],base[mm].matrix[7],base[mm].matrix[8]);

//sleep(1);

m--;

i++;

}

}

void leave()/\* 推理成功后退出程序之前要执行的函数，主要作用是输出结果\*/

{

n--;

while(base[n].father!=-1)

{result[m]=n;

m++;

n=base[n].father;}

result[m]=0;

result[m+1]='\0';

show();

//clrscr();

printf("\n\n\n\n\n\n\n\n\n\t\t\t\t搜索结束\n\n\n\n\n\n\n\n\n\n");

getch();

//exit(0);

}

int left(int x)/\*把下标为X的数组中的矩阵的空格左移\*/

{

int i,j;

char ch;

for(i=0;i<9;i++)

{if(base[x].matrix[i]==' ')

break;}

if(i==0|i==3|i==6|i==9)

{

return 0;

}

for(j=0;j<9;j++) {

base[n].matrix[j]=base[x].matrix[j];}

ch=base[n].matrix[i-1];

base[n].matrix[i-1]=base[n].matrix[i];

base[n].matrix[i]=ch;

base[n].operate='R';

base[n].extend='Y';

base[n].father=x;

base[x].extend='N';

n++;

if(match(i))

leave();

return 1;

}

int right(int x)/\*把下标为X的数组中的矩阵的空格右移\*/

{int i,j;

char ch;

for(i=0;i<9;i++)

{if(base[x].matrix[i]=='')

break;}

if(i==2|i==5|i==8|i==9)

{return 0;}

for(j=0;j<9;j++){

base[n].matrix[j]=base[x].matrix[j];}

ch=base[n].matrix[i+1];

base[n].matrix[i+1]=base[n].matrix[i];

base[n].matrix[i]=ch;

base[n].operate='L';

base[n].extend='Y';

base[n].father=x;

base[x].extend='N';

n++;

if(match(i))

leave();

return 1;}

int up(int x)/\*把下标为X的数组中的矩阵的空格上移\*/

{int i,j;

char ch;

for(i=0;i<9;i++)

{if(base[x].matrix[i]=='')

break;}

if(i==0|i==1|i==2|i==9)

{

return 0;}

for(j=0;j<9;j++){

base[n].matrix[j]=base[x].matrix[j];

ch=base[n].matrix[i-3];

base[n].matrix[i-3]=base[n].matrix[i];

base[n].matrix[i]=ch;

base[n].operate ='D';

base[n].extend='Y';

base[n].father=x;

base[x].extend='N';

n++;

if(match(i))

leave();

return 1;}

int down(int x)/\*把下标为X的数组中的矩阵的空格下移\*/

{int i,j;

char ch;

for(i=0;i<9;i++)

if(base[x].matrix[i]=='')

break;}

if(i==6|i==7|i==8|i==9){

return 0;}

for(j=0;j<9;j++){

base[n].matrix[j]=base[x].matrix[j];}

ch=base[n].matrix[i+3];

base[n].matrix[i+3]=base[n].matrix[i];

base[n].matrix[i]=ch;

base[n].operate='U';

base[n].extend='Y';

base[n].father=x;

base[x].extend='N';

n++;

if(match(i))

leave();

return 1;}

main()

{int i;

char a[20],b[20];

n=1;

//textcolor(LIGHTGREEN);

//clrscr();

/\*以下是输入初始和目标矩阵，并把输入的0转换为空格\*/

printf("Please input the start 9 chars:");

scanf("%s",a);

printf("Please input the end 9 chars:");

scanf("%s",b);

for(i=0;i<9;i++)

{if(a[i]=='0')

{

start[i]='';

continue;}

if(b[i]=='0')

{

end[i]='';

continue;}

start[i]=a[i];

end[i]=b[i];

start[9]='\0';

end[9]='\0';

for(i=0;i<9;i++)

base[0].matrix[i]=start[i];

base[0].operate='N';

base[0].extend='Y';

base[0].father=-1;

/\*以上是为第一个base数组元素赋值\*/

for(i=0;n<4000;i++)

{if(base[i].extend=='Y')

{if(base[i].operate=='L')

{right(i);up(i);down(i);}

if(base[i].operate=='R')

{left(i);up(i);down(i);}

if(base[i].operate=='U')

{left(i);right(i);down(i);}

if(base[i].operate=='D')

{left(i);right(i);up(i);}

if(base[i].operate=='N')

{left(i);right(i);up(i);down(i);

}

}

}

}