深度优先算法

**#include <windows.h>**

**#include "iostream.h"**

**#include "stdio.h"**

**#include "stdlib.h"**

**#include "time.h"**

**#include "string.h"**

**#include <queue>**

**#include <stack>**

**using namespace std;**

**const int N = 3;//3\*3图**

**enum Direction{None,Up,Down,Left,Right};//方向**

**static int n=0;**

**struct Map//图**

**{**

**int cell[N][N];//数码数组**

**Direction BelockDirec;//所屏蔽方向**

**int step;**

**struct Map \* Parent;//父节点**

**};**

**//打印图**

**void PrintMap(struct Map \*map)**

**{**

**cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;**

**for(int i=0;i<N;i++)**

**{**

**for(int j=0;j<N;j++)**

**{**

**cout<<map->cell[i][j]<<" ";**

**}**

**cout<<endl;**

**}**

**cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;**

**}**

**//移动图**

**struct Map \* MoveMap(struct Map \* map,Direction Direct,bool CreateNewMap)**

**{**

**struct Map \* NewMap;**

**//获取空闲格位置**

**int i,j;**

**for(i = 0; i < N; i++)**

**{**

**bool HasGetBlankCell = false;**

**for(j = 0; j < N; j++)**

**{**

**if(map->cell[i][j] == 0)**

**{**

**HasGetBlankCell = true;**

**break;**

**}**

**}**

**if(HasGetBlankCell)**

**break;**

**}**

**//移动数字**

**int t\_i = i,t\_j = j;**

**bool AbleMove = true;**

**switch(Direct)//判断沿direct所指方向移动数字是否被允许**

**{**

**case Down:**

**t\_i++;**

**if(t\_i >= N)**

**AbleMove=false;**

**break;**

**case Up:**

**t\_i--;**

**if(t\_i < 0)**

**AbleMove=false;**

**break;**

**case Left:**

**t\_j--;**

**if(t\_j < 0)**

**AbleMove=false;**

**break;**

**case Right:**

**t\_j++;**

**if(t\_j >= N)**

**AbleMove=false;**

**break;**

**};**

**if(!AbleMove)//不可以移动则返回原节点**

**{**

**return map;**

**}**

**if(CreateNewMap)**

**{**

**NewMap = new Map();**

**for(int x = 0; x < N; x++)**

**for(int y = 0; y < N; y++)**

**NewMap->cell[x][y] = map->cell[x][y];**

**}**

**else NewMap = map;**

**NewMap->cell[i][j] = NewMap->cell[t\_i][t\_j];**

**NewMap->cell[t\_i][t\_j] = 0;**

**return NewMap;**

**}**

**//初始化一个初始图**

**//由目标图生成初始图，保证可以获得结果**

**struct Map \* RandomMap(const struct Map \* map)**

**{**

**int M = 30;//随机移动图步数**

**struct Map \* NewMap;**

**NewMap = new Map();**

**memcpy(NewMap,map,sizeof(Map));**

**srand((unsigned)time(NULL));**

**for(int i = 0; i < M; i++)**

**{**

**int Direct = rand()%4;**

**NewMap = MoveMap(NewMap,(Direction)Direct,false);**

**}**

**return NewMap;**

**}**

**//初始图的另种生成方式，随机生成各位置的数**

**//此方式生成的图在有限次搜索中若深度超过5则多数无解**

**struct Map \* RandomMap()**

**{**

**int a[9];**

**struct Map \* NewMap;**

**NewMap = new Map();**

**srand((unsigned)time(NULL));**

**for(int k = 0; k < 9; k++)**

**{**

**bool Isre = false;**

**a[k] = rand()%9;**

**for (int l = 0; l < k; l++)**

**{**

**if (a[k] == a[l])**

**{**

**Isre = true;**

**break;**

**}**

**}**

**if(Isre)**

**{**

**k = k - 1;**

**continue;**

**}**

**}**

**for(int i = 0; i < N; i++)**

**{**

**for (int j = 0; j < N; j++)**

**{**

**NewMap->cell[i][j] = a[i\*3+j];**

**}**

**}**

**NewMap->Parent = NULL;**

**NewMap->BelockDirec = None;**

**return NewMap;**

**}**

**//判断是否搜索成功**

**bool IsSuccess(struct Map \* map,struct Map \* Target)**

**{**

**bool IsSuc = true;**

**for(int i = 0; i < N; i++)**

**{**

**for(int j = 0; j < N; j++)**

**{**

**if(map->cell[i][j] != Target->cell[i][j])**

**{**

**IsSuc = false;**

**break;**

**}**

**}**

**if(!IsSuc)**

**break;**

**}**

**return IsSuc;**

**}**

**struct Map \* DNF\_Search(struct Map \* begin,struct Map \* Target,int dm)**

**{**

**struct Map \* p1, \*p2,\*T=NULL;**

**stack<Map \*> OPEN;**

**stack<Map \*> CLOSED;**

**OPEN.push(begin);**

**do**

**{**

**p1=OPEN.top();**

**OPEN.pop();**

**if(IsSuccess(p1,Target))**

**{**

**T=p1;**

**return T;**

**}**

**if(p1->step==dm)**

**{**

**CLOSED.push(p1);**

**continue;**

**}**

**for (int i = 1; i <= 4; i++)**

**{**

**Direction Direct=(Direction)i;**

**if(Direct == p1->BelockDirec)//跳过屏蔽方向**

**continue;**

**p2 = MoveMap(p1,Direct,true);**

**if(p2 != p1)//数码是否可以移动**

**{**

**p2->Parent = p1;**

**p2->step = p1->step + 1;**

**switch(Direct)//设置屏蔽方向,防止往回推**

**{**

**case Up:**

**p2->BelockDirec = Down;**

**break;**

**case Down:**

**p2->BelockDirec = Up;**

**break;**

**case Left:**

**p2->BelockDirec = Right;**

**break;**

**case Right:**

**p2->BelockDirec = Left;**

**break;**

**}**

**OPEN.push(p2);**

**n++;**

**}**

**}**

**}while(!OPEN.empty());**

**return T;**

**}**

**void main()**

**{**

**Map Target;**

**Map \*begin,\*T;**

**int step=1;**

**//设定目标图 [1 2 3],[8 0 4],[7 6 5]**

**Target.cell[0][0] = 1;**

**Target.cell[0][1] = 2;**

**Target.cell[0][2] = 3;**

**Target.cell[1][0] = 8;**

**Target.cell[1][1] = 0;**

**Target.cell[1][2] = 4;**

**Target.cell[2][0] = 7;**

**Target.cell[2][1] = 6;**

**Target.cell[2][2] = 5;**

**Target.step = 0;**

**/\***

**begin = new Map();**

**begin->cell[0][0] = 2;**

**begin->cell[0][1] = 8;**

**begin->cell[0][2] = 3;**

**begin->cell[1][0] = 1;**

**begin->cell[1][1] = 0;**

**begin->cell[1][2] = 4;**

**begin->cell[2][0] = 7;**

**begin->cell[2][1] = 6;**

**begin->cell[2][2] = 5;**

**Target.step = 0;\*/**

**begin = RandomMap();**

**// begin = RandomMap(&Target);**

**begin->cell[0][0]=6;**

**begin->cell[0][1]=3;**

**begin->cell[0][2]=2;**

**begin->cell[1][0]=1;**

**begin->cell[1][1]=4;**

**begin->cell[1][2]=8;**

**begin->cell[2][0]=7;**

**begin->cell[2][1]=5;**

**begin->cell[2][2]=0;**

**/\* begin->cell[0][0]=2;**

**begin->cell[0][1]=1;**

**begin->cell[0][2]=6;**

**begin->cell[1][0]=4;**

**begin->cell[1][1]=0;**

**begin->cell[1][2]=8;**

**begin->cell[2][0]=7;**

**begin->cell[2][1]=5;**

**begin->cell[2][2]=3;**

**\*/**

**//begin = RandomMap(&Target);**

**begin->Parent = NULL;**

**begin->BelockDirec = None;**

**begin->step = 0;**

**cout<<"目标图:"<<endl;**

**PrintMap(&Target);**

**cout<<"起始图:"<<endl;**

**PrintMap(begin);**

**//图搜索**

**int dm;//搜索路劲深度**

**cout<<"搜索路径深度:";**

**cin>>dm;**

**T=DNF\_Search(begin,&Target,dm);**

**//打印**

**if(T != NULL)**

**{**

**//把路径倒序**

**Map \*p=T;**

**stack<Map \*> Stack1;**

**while(p->Parent != NULL)**

**{**

**Stack1.push(p);**

**p = p->Parent;**

**}**

**Stack1.push(begin);**

**cout<<"搜索结果:"<<endl;**

**while(!Stack1.empty())**

**{**

**cout<<"第"<<step++<<"步："<<endl;**

**PrintMap(Stack1.top());**

**Stack1.pop();**

**}**

**cout<<"\n完成!";**

**cout<<"生成节点的个数："<<n<<endl;**

**}**

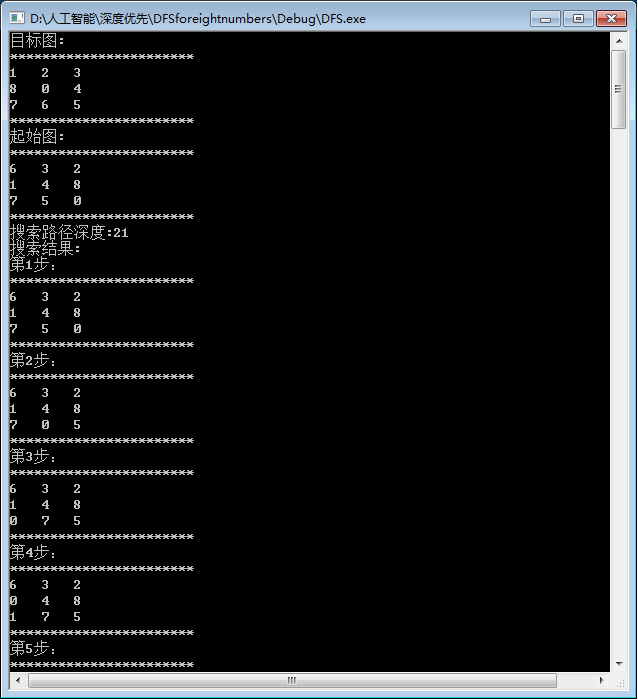
**else cout<<"达到了指定的搜索路径长度未搜到目标"<<endl;**

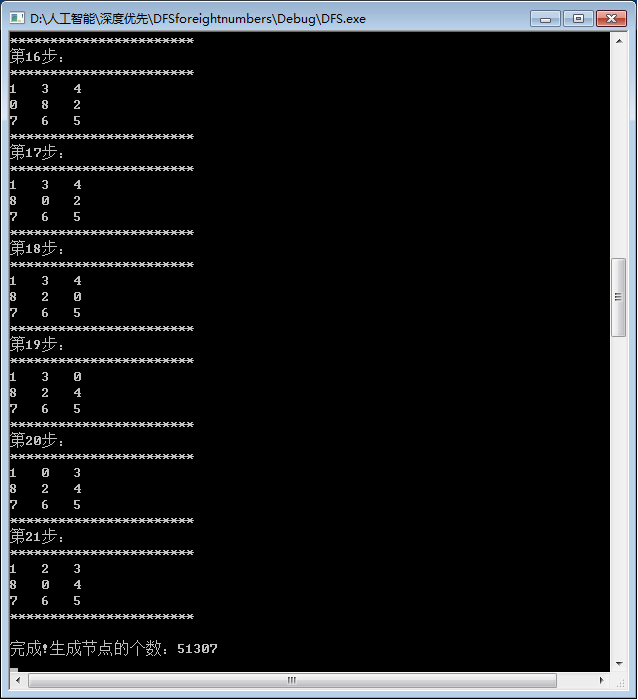
**int input;**

**cin>>input;**

**}**

**程序截图如下：**

****

****