**图的深度优先遍历**

图的深度优先遍历的思想：假设初始状态是图中所有顶点均未被访问，则从某个顶点v出发，首先访问该顶点，然后依次从它的各个未被访问的邻接点出发深度优先搜索遍历图，直至图中所有和v有路径相通的顶点都被访问到。 若此时尚有其他顶点未被访问到，则另选一个未被访问的顶点作起始点，重复上述过程，直至图中所有顶点都被访问到为止。

图的深度优先遍历特点是：选定一个出发点后进行遍历，能前进则前进，若不能前进，回退一步再前进，或再回退一步后继续前进。依此重复，直到所有与选定点相通的所有顶点都被遍历。

#include <iostream>  
#include <stdlib.h>  
#include <stdio.h>  
using namespace std;  
int visited[20];  
typedef struct ArcNode{  
 int adgvex;  
 ArcNode \*nextarc;  
 int \*info;  
}ArcNode;  
typedef struct Vnode{  
 char data;  
 ArcNode \*firstarc;  
}Vnode,AdjList[20];  
typedef struct{  
 AdjList vertices;  
 int vexnum,arcnum;  
 int kind;  
}ALGraph;  
int LocateVex(ALGraph &G,char vp){  
 int k=0;  
 for(k=0;k<G.vexnum;k++)  
 {  
 if(G.vertices[k].data==vp) return k;  
 }  
 return -1;  
}  
void CreatALGraph (ALGraph &G)  
{  
 char e1,e2=0;  
 int k,l=0;  
 int m=0;  
 ArcNode \*s;  
 cin>>G.arcnum>>G.vexnum;  
 for(int i=0;i<G.vexnum;i++)  
 {  
 cin>>G.vertices[i].data;  
 G.vertices[i].firstarc=NULL;  
 }  
 //s=(ArcNode \*)malloc (sizeof(ArcNode));  
 for(k=0;k<G.arcnum;k++)  
 {  
 cin>>e1>>e2;  
 s=(ArcNode \*)malloc (sizeof(ArcNode));  
 l=LocateVex(G,e1);  
 m=LocateVex(G,e2);  
 s->adgvex=m;  
 s->nextarc=G.vertices[l].firstarc;  
 G.vertices[l].firstarc=s;  
 }  
}  
void OutputALGraph (ALGraph &G)  
{  
 int i=0;  
 for(i=0;i<G.vexnum;i++)  
 {  
 ArcNode \*s;  
 printf("%d:%c",i,G.vertices[i].data);  
 s=G.vertices[i].firstarc;  
 while(s!=NULL)  
 {  
 printf("%4d",s->adgvex);  
 s=s->nextarc;  
 }  
 printf("\n")；  
 }  
}  
void DFS(ALGraph &G,int v){  
 ArcNode \*p;  
 if(visited[v]==0)  
 {

printf("%c",G.vertices[v].data);  
 visited[v]=1;  
 p=G.vertices[v].firstarc;  
 while (p!=NULL)  
 {  
 if(!visited[p->adgvex])  
 {DFS(G,p->adgvex);}  
 p=p->nextarc;  
 }

}

}  
void DFS\_Traverse\_Graph(ALGraph &G,int t)  
{  
 for(int v=0;v<G.vexnum;v++)  
 {visited[v]=0;}  
 DFS(G,t);  
 for(int v=0;v<G.vexnum;v++)  
 {if(visited[v]==0)  
 {DFS(G,v);}  
 }  
}  
void main()  
{  
 ALGraph G;  
 char m;  
 int t=0;  
 CreatALGraph(G);  
 OutputALGraph(G);  
 cin>>m;  
 t=LocateVex(G,m);  
 DFS\_Traverse\_Graph(G,t);  
}

