#include<stdio.h>

#include<stdlib.h>

#define MAX 20

typedef int VexType;

typedef VexType Mgraph[MAX][MAX];/\* Mgraph是二维数组类型标识符 \*/

/\* 函数声明表示 \*/

void creat\_mg(Mgraph G);

void out\_mg(Mgraph G);  /\* G1是邻接矩阵的二维数组名 \*/

void printDegree(Mgraph G);

Mgraph G1;

int n, e, v0;

/\* 主函数 \*/

int main(){

    creat\_mg(G1);

    out\_mg(G1);

    printDegree(G1);

}

 /\* 建立邻接矩阵 \*/

void creat\_mg(Mgraph G){

    int i, j, k;

    printf("\n  n,e=?");

    scanf("%d%d", &n, &e);/\* 输入顶点数n,e \*/

    for (int i =1; i <= n; i++){

        for (int j = 0; j <= n;j++){

            G[i][j] = 0;

        }

    }

    for (int k = 1; k <= e;k++){

        printf("\n  vi,vj=?");

        scanf("%d%d", &i, &j);

        G[i][j] = 1;

        G[j][i] = 1;

    }

}

void out\_mg(Mgraph G){

    int i, j, k;

    char ch;

    for (i = 1; i <= n;i++){

        printf("\n");

        for (j = 1; j <= n;j++){

            printf("%5d", G[i][j]);

        }

    }

    for (i = 1; i <= n;i++){

        for (j = 1; j <= n; j++)

        {

            if(G[i][j]==1){

                printf("\n 存在边< %d,%d >",i, j);

            }

        }

    }

}

void printDegree(Mgraph G){

    int degree;

    for (int i = 1; i <= n; i++) {

        degree = 0;

        for (int j = 1; j <= n; j++) {

            if (G[i][j] == 1) {

                degree++;

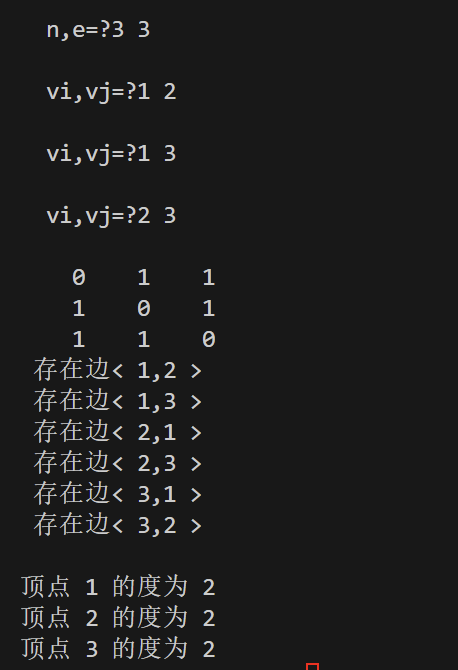
            }

        }

        printf("\n顶点 %d 的度为 %d", i, degree);

    }

}



//无向图的存储-邻接矩阵

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4 6

A B C D

A B

A C

A D

B C

B D

C D

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <string.h>

#define N (100 + 5)

#define INF 0x3f3f3f3f

typedef char VexType;   //本题，顶点类型为字符类型

typedef int EdgeType;   //本题，邻接矩阵类型为整型

//图结构-顶点和边 G = <V,E>

typedef struct{

    int n,m;            //n个顶点m条边

    VexType vex[N];     //顶点数组

    EdgeType edge[N][N];//邻接矩阵

}adjGraph;

adjGraph createGraph(); //邻接矩阵建图

void print(adjGraph g); //输出图的信息(顶点、邻接矩阵）

void printDegree(adjGraph g);//输出图中每个顶点的度数

int main()

{

    //1.建图

    adjGraph g = createGraph();

    //2.输出图的信息

    print(g);

    printDegree(g);

    return 0;

}

//邻接矩阵建图

adjGraph createGraph()

{

    adjGraph g;

    //0.邻接矩阵初始化为0

    memset(g.edge,0,sizeof(g.edge));

    scanf("%d%d",&g.n,&g.m);    //输入顶点数和边数

    getchar();      //吸收换行符

    //1.输入n个顶点

    for(int i = 0;i < g.n;i++){

        scanf("%c ",&g.vex[i]);

    }

    //2.输入m条边，按照邻接矩阵存图

    for(int i = 0;i < g.m;i++){

        char v1,v2;

        scanf("\n%c %c",&v1,&v2);   //读入当前边的2个顶点

        int n1 = v1 - 'A',n2 = v2 - 'A';

        //无向图，邻接矩阵对应的n1行n2列和n2行n1列都赋值为1

        g.edge[n1][n2] = g.edge[n2][n1] = 1;

    }

    return g;

}

//输出图的信息(顶点、邻接矩阵）

void print(adjGraph g)

{

    printf("图有%d个顶点，%d条边\n",g.n,g.m);

    printf("图的顶点是：");

    for(int i = 0;i < g.n;i++){

        printf("%c ",g.vex[i]);

    }

    //输出邻接矩阵

    printf("\n图的邻接矩阵是：\n");

    for(int i = 0;i < g.n;i++){

        for(int j = 0;j < g.n;j++){

            printf("%4d",g.edge[i][j]);

        }

        printf("\n");

    }

}

//输出图中每个顶点的度数

void printDegree(adjGraph g)

{

    int degree;

    printf("\n图中每个顶点的度数如下：\n");

    for (int i = 0; i < g.n; i++)

    {

        degree = 0;

        for (int j = 0; j < g.n; j++)

        {

            // 无向图中，顶点的度数等于其邻接矩阵对应行（或列）中值为1的元素个数

            if (g.edge[i][j] == 1)

            {

                degree++;

            }

        }

        printf("顶点 %c 的度数为：%d\n", g.vex[i], degree);

    }

}

