实验09图及应用（一）

实验周次：第16周 学时：2学时 地点：学院机房

学号：3190707121 姓名：武新纪 班级：人工智能191 序号：21

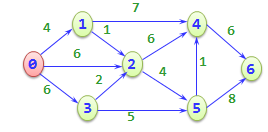
提示：请务必填写 以上个人信息。

实验每题分，共计题。

**【实验目的】**

1.领会图的两种主要存储结构和基于图的基本运算算法设计。

**【实验内容】**

****

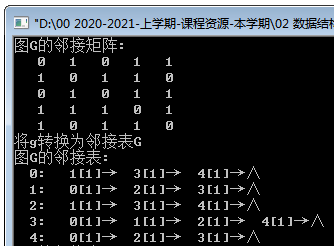
**针对以上带权有向图G。**

实验1.编写一个程序graph.cpp,设计带权图的邻接矩阵与邻接表的创建和输出运算，并在此基础上设计一个主程序exp8-1.cpp完成以下功能。

（1）建立如上图所示的有向图G的邻接矩阵，并输出（输出图的邻接矩阵二维数组形式），格式如下：

（2）销毁图G的邻接矩阵。

（3）建立如上图所示的有向图G的邻接表，并输出（输出各顶点及其邻边），格式如下：



（4）销毁图G的邻接表。

**程序：**

#include <stdio.h>

#include <malloc.h>

#define INF 32767 //定义∞

#define MAXV 100  //最大顶点个数

typedef char InfoType;

//以下定义邻接矩阵类型

typedef struct

{

    int no;        //顶点编号

    InfoType info; //顶点其他信息

} VertexType;      //顶点类型

typedef struct

{

    int edges[MAXV][MAXV]; //邻接矩阵数组

    int n, e;              //顶点数，边数

    VertexType vexs[MAXV]; //存放顶点信息

} MatGraph;                //完整的图邻接矩阵类型

//以下定义邻接表类型

typedef struct ANode

{

    int adjvex;            //该边的邻接点编号

    struct ANode \*nextarc; //指向下一条边的指针

    int weight;            //该边的相关信息，如权值（用整型表示）

} ArcNode;                 //边节点类型

typedef struct Vnode

{

    InfoType info;     //顶点其他信息

    int count;         //存放顶点入度,仅仅用于拓扑排序

    ArcNode \*firstarc; //指向第一条边

} VNode;               //邻接表头节点类型

typedef struct

{

    VNode adjlist[MAXV]; //邻接表头节点数组

    int n, e;            //图中顶点数n和边数e

} AdjGraph;              //完整的图邻接表类型

void CreateMat(MatGraph &g, int A[MAXV][MAXV], int n, int e) //创建图的邻接矩阵

{

    int i, j;

    g.n = n;

    g.e = e;

    for (i = 0; i < g.n; i++)

        for (j = 0; j < g.n; j++)

            g.edges[i][j] = A[i][j];

}

void DispMat(MatGraph g) //输出邻接矩阵g

{

    int i, j;

    for (i = 0; i < g.n; i++)

    {

        for (j = 0; j < g.n; j++)

            if (g.edges[i][j] != INF)

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

            else

                printf("%-4s", "∞");

        printf("\n");

    }

}

void CreateAdj(AdjGraph \*&G, int A[MAXV][MAXV], int n, int e) //创建图的邻接表

{

    int i, j;

    ArcNode \*p;

    G = (AdjGraph \*)malloc(sizeof(AdjGraph));

    for (i = 0; i < n; i++) //给邻接表中所有头节点的指针域置初值

        G->adjlist[i].firstarc = NULL;

    for (i = 0; i < n; i++) //检查邻接矩阵中每个元素

        for (j = n - 1; j >= 0; j--)

            if (A[i][j] != 0 && A[i][j] != INF) //存在一条边

            {

                p = (ArcNode \*)malloc(sizeof(ArcNode)); //创建一个节点p

                p->adjvex = j;

                p->weight = A[i][j];

                p->nextarc = G->adjlist[i].firstarc; //采用头插法插入节点p

                G->adjlist[i].firstarc = p;

            }

    G->n = n;

    G->e = n;

}

void DispAdj(AdjGraph \*G) //输出邻接表G

{

    ArcNode \*p;

    for (int i = 0; i < G->n; i++)

    {

        p = G->adjlist[i].firstarc;

        printf("%3d: ", i);

        while (p != NULL)

        {

            printf("%3d[%d]→", p->adjvex, p->weight);

            p = p->nextarc;

        }

        printf("∧\n");

    }

}

void DestroyAdj(AdjGraph \*&G) //销毁图的邻接表

{

    ArcNode \*pre, \*p;

    for (int i = 0; i < G->n; i++) //扫描所有的单链表

    {

        pre = G->adjlist[i].firstarc; //p指向第i个单链表的首节点

        if (pre != NULL)

        {

            p = pre->nextarc;

            while (p != NULL) //释放第i个单链表的所有边节点

            {

                free(pre);

                pre = p;

                p = p->nextarc;

            }

            free(pre);

        }

    }

    free(G); //释放头节点数组

}

int main()

{

    MatGraph g;

    AdjGraph \*G;

    int A[MAXV][MAXV] = {

        {0, 4, 6, 6, INF, INF, INF},

        {INF, 0, 1, INF, 7, INF, INF},

        {INF, INF, 0, INF, 6, 4, INF},

        {INF, INF, 2, 0, INF, 5, INF},

        {INF, INF, INF, INF, 0, INF, 6},

        {INF, INF, INF, INF, 1, 0, 8},

        {INF, INF, INF, INF, INF, INF, 0}

    };

    int n = 7, e = 12;

    CreateMat(g, A, n, e);

    printf("(1)图G的邻接矩阵:\n");

    DispMat(g);

    CreateAdj(G, A, n, e);

    printf("(2)图G的邻接表:\n");

    DispAdj(G);

    printf("(3)销毁图G的邻接表\n");

    DestroyAdj(G);

    return 0;

}

**运行结果：**

****