数据结构实验报告五

学号：201930310012 姓名：张金宇

#include<string.h>

#include<ctype.h>

#include<malloc.h> /\* malloc()等 \*/

#include<limits.h> /\* INT\_MAX等 \*/

#include<stdio.h> /\* EOF(=^Z或F6),NULL \*/

#include<stdlib.h> /\* atoi() \*/

#include<io.h> /\* eof() \*/

#include<math.h> /\* floor(),ceil(),abs() \*/

#include<process.h> /\* exit() \*/

#include<string.h>

/\* 函数结果状态代码 \*/

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASIBLE -1

/\* #define OVERFLOW -2 因为在math.h中已定义OVERFLOW的值为3,故去掉此行 \*/

typedef int Status; /\* Status是函数的类型,其值是函数结果状态代码，如OK等 \*/

typedef int Boolean; /\* Boolean是布尔类型,其值是TRUE或FALSE \*/

#define INFINITY INT\_MAX /\* 用整型最大值代替∞ \*/

#define MAX\_VERTEX\_NUM 20 /\* 最大顶点个数 \*/

typedef enum { DG, DN, AG, AN }GraphKind; /\* {有向图,有向网,无向图,无向网} \*/

#define MAX\_NAME 5 /\* 顶点字符串的最大长度+1 \*/

#define MAX\_INFO 20 /\* 相关信息字符串的最大长度+1 \*/

typedef int VRType;

typedef char InfoType;

typedef char VertexType[MAX\_NAME];

typedef struct

{

VRType adj; /\* 顶点关系类型。对无权图，用1(是)或0(否)表示相邻否； \*/

/\* 对带权图，c则为权值类型 \*/

InfoType \*info; /\* 该弧相关信息的指针(可无) \*/

}ArcCell, AdjMatrix[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM];

typedef struct

{

VertexType vexs[MAX\_VERTEX\_NUM]; /\* 顶点向量 \*/

AdjMatrix arcs; /\* 邻接矩阵 \*/

int vexnum, arcnum; /\* 图的当前顶点数和弧数 \*/

GraphKind kind; /\* 图的种类标志 \*/

}MGraph;

int LocateVex(MGraph G, VertexType u)

{ /\* 初始条件:图G存在,u和G中顶点有相同特征 \*/

/\* 操作结果:若G中存在顶点u,则返回该顶点在图中位置;否则返回-1 \*/

int i;

for (i = 0; i<G.vexnum; ++i)

if (strcmp(u, G.vexs[i]) == 0)

return i;

return -1;

}

Status CreateFAG(MGraph \*G)

{ /\* 采用数组(邻接矩阵)表示法,由文件构造没有相关信息的无向图G \*/

int i, j, k;

char filename[13];

VertexType va, vb;

FILE \*graphlist;

printf("请输入数据文件名(f7-1.dat)：");

scanf("%s", filename);

graphlist = fopen(filename, "r");

fscanf(graphlist, "%d", &(\*G).vexnum);

fscanf(graphlist, "%d", &(\*G).arcnum);

for (i = 0; i<(\*G).vexnum; ++i) /\* 构造顶点向量 \*/

fscanf(graphlist, "%s", (\*G).vexs[i]);

for (i = 0; i<(\*G).vexnum; ++i) /\* 初始化邻接矩阵 \*/

for (j = 0; j<(\*G).vexnum; ++j)

{

(\*G).arcs[i][j].adj = 0; /\* 图 \*/

(\*G).arcs[i][j].info = NULL; /\* 没有相关信息 \*/

}

for (k = 0; k<(\*G).arcnum; ++k)

{

fscanf(graphlist, "%s%s", va, vb);

i = LocateVex(\*G, va);

j = LocateVex(\*G, vb);

(\*G).arcs[i][j].adj = (\*G).arcs[j][i].adj = 1; /\* 无向图 \*/

}

fclose(graphlist);

(\*G).kind = AG;

return OK;

}

Status CreateDG(MGraph \*G)

{ /\* 采用数组(邻接矩阵)表示法,构造有向图G \*/

int i, j, k, l, IncInfo;

char s[MAX\_INFO], \*info;

VertexType va, vb;

printf("请输入有向图G的顶点数,弧数,弧是否含其它信息(是:1,否:0): ");

scanf("%d,%d,%d", &(\*G).vexnum, &(\*G).arcnum, &IncInfo);

printf("请输入%d个顶点的值(<%d个字符):\n", (\*G).vexnum, MAX\_NAME);

for (i = 0; i<(\*G).vexnum; ++i) /\* 构造顶点向量 \*/

scanf("%s", (\*G).vexs[i]);

for (i = 0; i<(\*G).vexnum; ++i) /\* 初始化邻接矩阵 \*/

for (j = 0; j<(\*G).vexnum; ++j)

{

(\*G).arcs[i][j].adj = 0; /\* 图 \*/

(\*G).arcs[i][j].info = NULL;

}

printf("请输入%d条弧的弧尾 弧头(以空格作为间隔): \n", (\*G).arcnum);

for (k = 0; k<(\*G).arcnum; ++k)

{

scanf("%s%s%\*c", va, vb); /\* %\*c吃掉回车符 \*/

i = LocateVex(\*G, va);

j = LocateVex(\*G, vb);

(\*G).arcs[i][j].adj = 1; /\* 有向图 \*/

if (IncInfo)

{

printf("请输入该弧的相关信息(<%d个字符): ", MAX\_INFO);

gets(s);

l = strlen(s);

if (l)

{

info = (char\*)malloc((l + 1)\*sizeof(char));

strcpy(info, s);

(\*G).arcs[i][j].info = info; /\* 有向 \*/

}

}

}

(\*G).kind = DG;

return OK;

}

Status CreateFNG(MGraph \*G)

{

return OK;

}

void main()

{

}

#include<stdlib.h>

#define MAXVEX 100

#define INFINITY 65535

typedef char VertexType //顶点的数据类型

typedef int EdgeType // 权值得类型

typedef struct {

VertexType vexs[MAXTEX];//顶点表

EdgeType arc[MAXTEX][MAXTEX];//邻接矩阵

int NumOfVertex, NumOfEdge; // 图中的边的数量和顶点的数量

}MyGraph;

void CreatGraph(MyGraph \* g) {

int i = 0, j = 0, k = 0, l = 0;

printf("请输入顶点的数量和边的数量 : ");

scanf("%d %d", &(g->NumOfVertex), &(g->NumOfEdge));

for (i = 0, i < (g->NumOfVertex), i++) {

scanf("%c", &(g->vexs[i]));

}

for (i = 0; i < (g->NumOfVertex); i++) {

for (j = 0; j < (g->NumOfVertex); j++) {

g->arc[i][j] = INFINITY;

}

}

for (k = 0; k < (g->NumOfEdge); k++) {

printf("输入边(vi , vj)上的下标 i , j 和 权");

scanf("%d %d %d ", &i, &j, &k);

g->arc[i][j] = k;

g->arc[i][j] = g->arc[j][i];//因为是无向边 , 矩阵对称

}

}

Status CreateFNG(MGraph \*G)

{

int i, j, k, l, IncInfo;

char s[MAX\_INFO], \*info;

VertexType va, vb;

printf("请输入有向图G的顶点数,弧数,弧是否含其它信息(是:1,否:0): ");

scanf("%d,%d,%d", &(\*G).vexnum, &(\*G).arcnum, &IncInfo);

printf("请输入%d个顶点的值(<%d个字符):\n", (\*G).vexnum, MAX\_NAME);

for (i = 0; i<(\*G).vexnum; ++i) /\* 构造顶点向量 \*/

scanf("%s", (\*G).vexs[i]);

for (i = 0; i<(\*G).vexnum; ++i) /\* 初始化邻接矩阵 \*/

for (j = 0; j<(\*G).vexnum; ++j)

{

(\*G).arcs[i][j].adj = 0; /\* 图 \*/

(\*G).arcs[i][j].info = NULL;

}

printf("请输入%d条弧的弧尾 弧头(以空格作为间隔): \n", (\*G).arcnum);

for (k = 0; k<(\*G).arcnum; ++k)

{

scanf("%s%s%\*c", va, vb); /\* %\*c吃掉回车符 \*/

i = LocateVex(\*G, va);

j = LocateVex(\*G, vb);

(\*G).arcs[i][j].adj = 1; /\* 有向图 \*/

if (IncInfo)

{

printf("请输入该弧的相关信息(<%d个字符): ", MAX\_INFO);

gets(s);

l = strlen(s);

if (l)

{

info = (char\*)malloc((l + 1)\*sizeof(char));

strcpy(info, s);

(\*G).arcs[i][j].info = info; /\* 有向 \*/

}

}

}

(\*G).kind = DG;

return OK; \

][

}

Status CreateAN(MGraph \*G)

{

int i, j, k, l, IncInfo;

int data;

char s[MAX\_INFO], \*info;

VertexType va, vb;

printf("请输入无向网G的顶点数,弧数,弧是否含其它信息(是:1,否:0): ");

scanf("%d,%d,%d", &(\*G).vexnum, &(\*G).arcnum, &IncInfo);

printf("请输入%d个顶点的值(<%d个字符):\n", (\*G).vexnum, MAX\_NAME);

for (i = 0; i<(\*G).vexnum; ++i) /\* 构造顶点向量 \*/

scanf("%s", (\*G).vexs[i]);

for (i = 0; i<(\*G).vexnum; ++i) /\* 初始化邻接矩阵 \*/

for (j = 0; j<(\*G).vexnum; ++j)

{

(\*G).arcs[i][j].adj = INT\_MAX; /\* 网 \*/

(\*G).arcs[i][j].info = NULL;

}

/\*for(k = 0 ; k < (g->NumOfEdge) ; k++){

printf("输入边(vi , vj)上的下标 i , j 和 权");

scanf("%d %d %d " , &i , &j , &k);

g->arc[i][j] = k;

g->arc[i][j] = g->arc[j][i];

for (k = 0; k<(\*G).arcnum; ++k)

{

scanf("%s%s%\*c", va, vb);

i = LocateVex(\*G, va);

j = LocateVex(\*G, vb);

scanf("%d", data);

(\*G).arcs[i][j].adj = (\*G).arcs[j][i].adj = data; // 无向网

}

(\*G).kind = AN;

return OK;

}

1. 这个程序的实现关键在于邻接矩阵的构建，可以根据有向网和无向图来写无向网的邻接矩阵。
2. 邻接矩阵是对称阵，所以只要构建一半就行。
3. 无向网的info可以有也可以没有，我在这里没写info。
4. 在这个程序中，有多个typedef，要理解每个typedef的意义，以便更好的使用它。